

Systeme de gestion de version de code source : « Git »

Module : Base de la Conception Orientée-Objet
DUT Informatique, Semestre 2

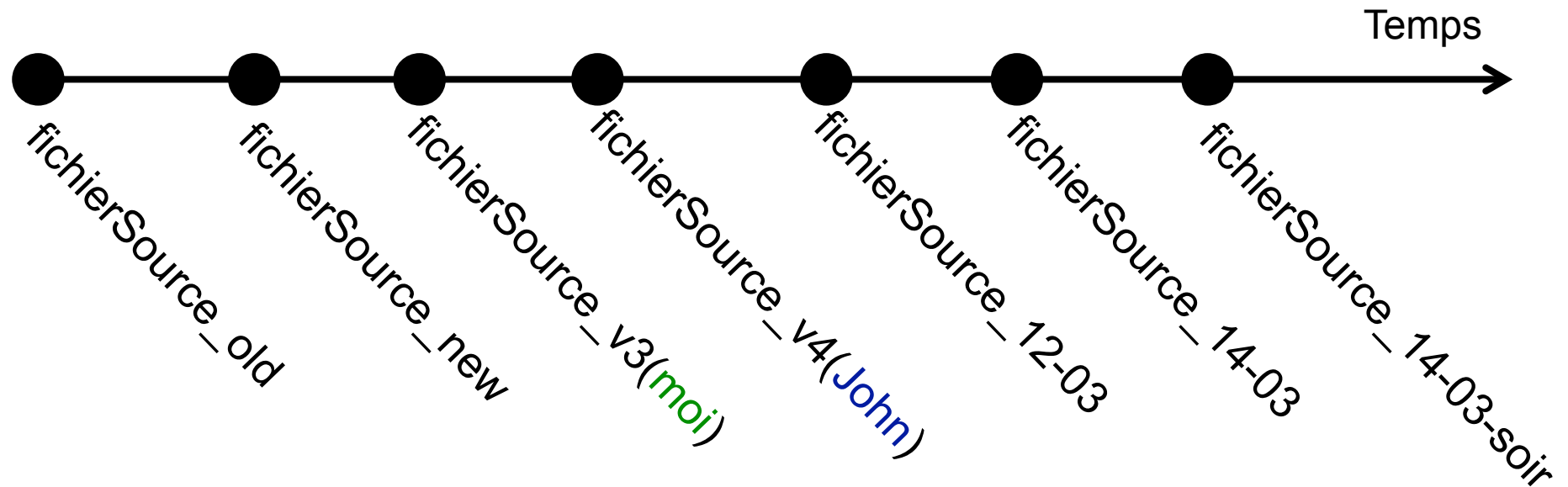
Mourad Ouziri

mourad.ouziri@parisdescartes.fr

IUT de Paris Descartes

Besoin d'un outil de suivi des versions de code source

- Cycle de vie d'un fichier de code : création, multiples modifications



- Le code fonctionnait hier mais plus après les modifications d'aujourd'hui !
- Comment revenir à la dernière version qui marchait ou à une version plus antérieure ?
- Qui a apporté un tel changement ?
- ...
- Besoin de revenir en arrière ! Tracer l'historique...

Besoin d'un outil de suivi de versions de code source

- Besoin de :
 - Travailler à plusieurs sur le même fichier de code
 - Garder l'historique des modifications
 - Revenir à une version antérieure (repartir d'une version qui marche !)
 - Caractériser chaque version (changements, date, etc.)
 - Identifier celui qui a généré une version (modification) de code
- Outil de gestion de code source (VCS – Version Control System) :
 - Outil de développement collaboratif (partage)
 - Outil gestion de versions de code source
 - Outil de sauvegarde et de restauration de code
- Catégorie : outil de gestion (optimisation) de projet

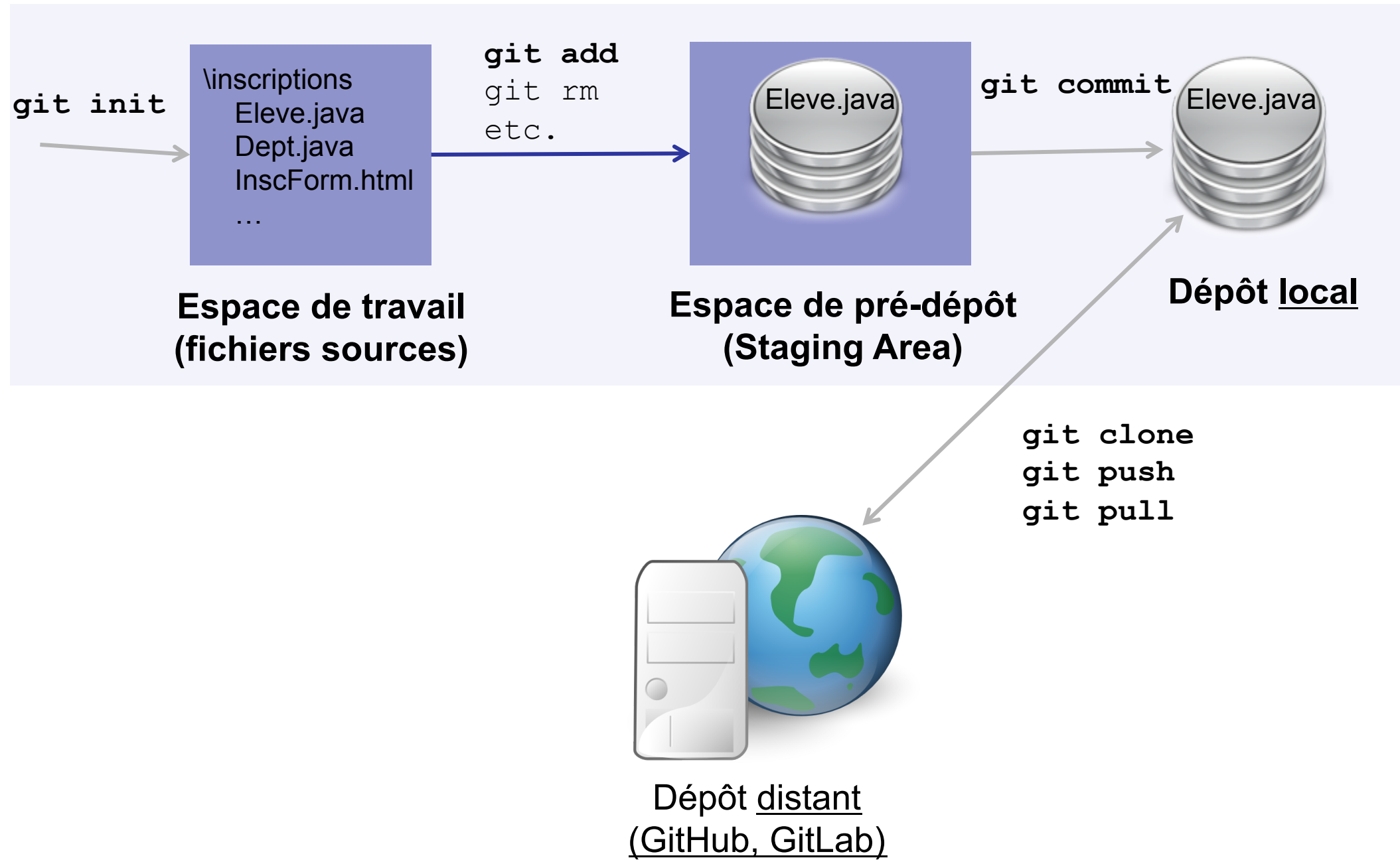
GIT : un outil de suivi de versions de code

- Git : un outil de gestion de code source :
 - Open source
 - Hébergement de projets (de code source) gratuit : GitHub / GitLab
 - Plusieurs outils gratuits
 - Distribué sur les machines des développeurs
- Support de protocoles « distants » :
 - HTTP(S) : `http://serveur/projet.git`
 - SSH : `user@serveur:projet.git`
 - GIT : `git://serveur/projet.git`
 - En local : `z:\mesprojets\projet.git`

GIT : un outil de suivi de versions de code

- GitHub : hébergement distant accessible par un navigateur Web
 - Git est en ligne de commande, GitHub en interface graphique Web
 - Offre plusieurs services de collaboration (wiki, outils de planification de tâches, etc.)
 - Les projets sont par défaut publics (sinon service payant)
 - Possibilité de récupérer (cloner) tous les projets, suivre/follow des projets et développeurs, poster des commentaires, etc.
- GitLab : hébergement sur serveur local et accessible en réseau
 - Pratique pour les projets internes
 - Celui installé ici à l'IUT : <http://gitlab.iutparis.local>

L'architecture et les espaces Git



Git : les espaces

- **Espace de travail (workspace)** : répertoire où se trouvent les fichiers de code source (répertoire de développement). Répertoire de travail du développeur
- **Le dépôt (repository)** : espace contenant les versions validées (par des opérations de « commit ») des fichiers de code source
- **Espace de pré-dépôt ou l'index (staging area)** : espace intermédiaire où sont déposées les versions de avant publication/ validation (commit) et mise en dépôt

Git : initialisation d'un projet/dépôt

- Configurer l'identité du développeur (qui a produit un code ?)

```
$ git config --global user.name "Mourad Ouziri"
```

```
$ git config --global user.email mourad.ouzi@parisdescartes.fr
```

Pour voir les configurations actuelles : **\$git config -l**

- Initialiser un espace de travail local (répertoire de code source) : le mettre sous contrôle de Git

```
$ cd z:\...\inscriptions
```

```
$ git init
```

git init



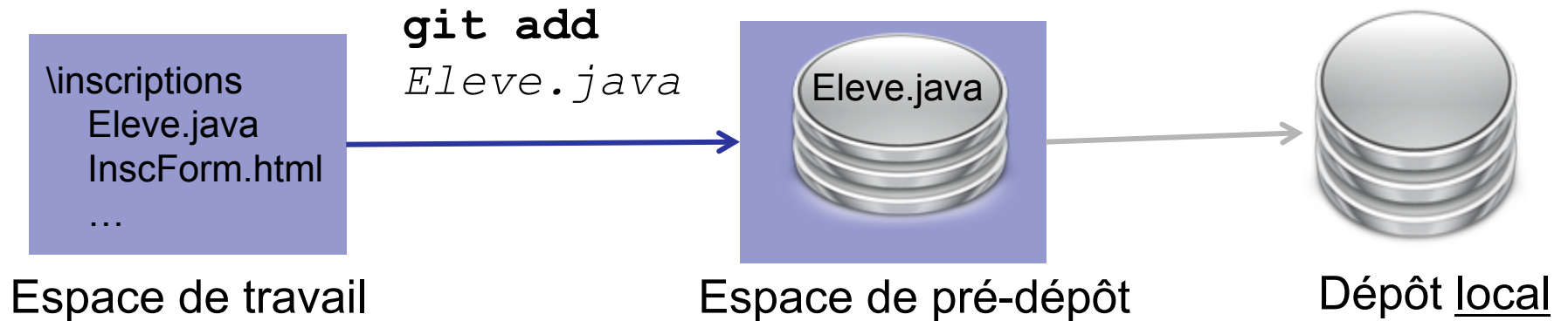
\inscriptions
Eleve.java
InscForm.html
...

Espace de travail

Git : ajout d'un fichier dans le pré-dépôt

- Ajouter un fichier à l'espace de pré-dépôt (prépare un commit) :

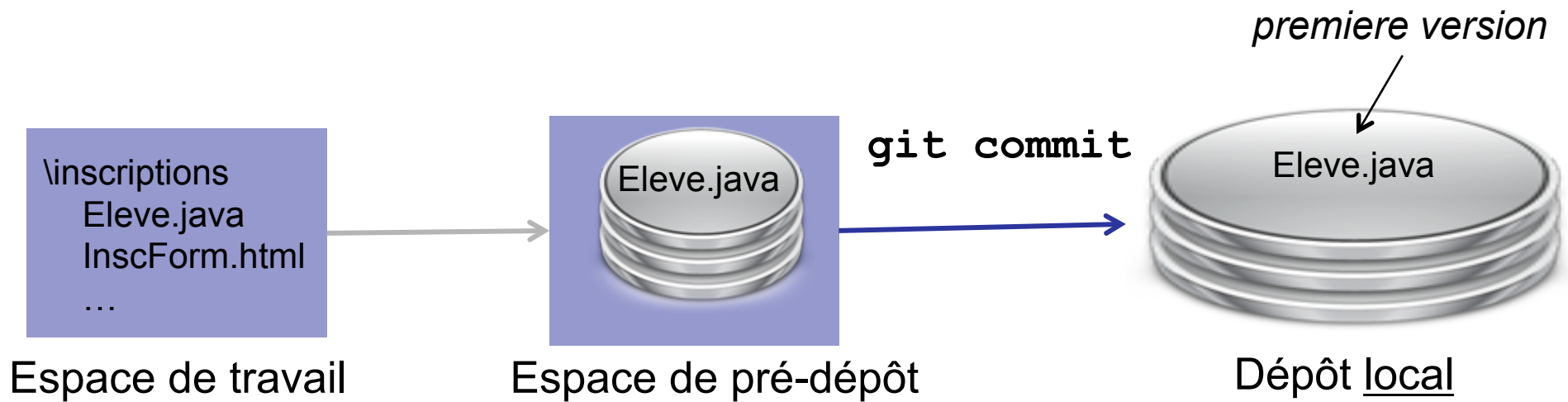
\$ **git add** *Eleve.java*



Git : validation (commit) d'une version

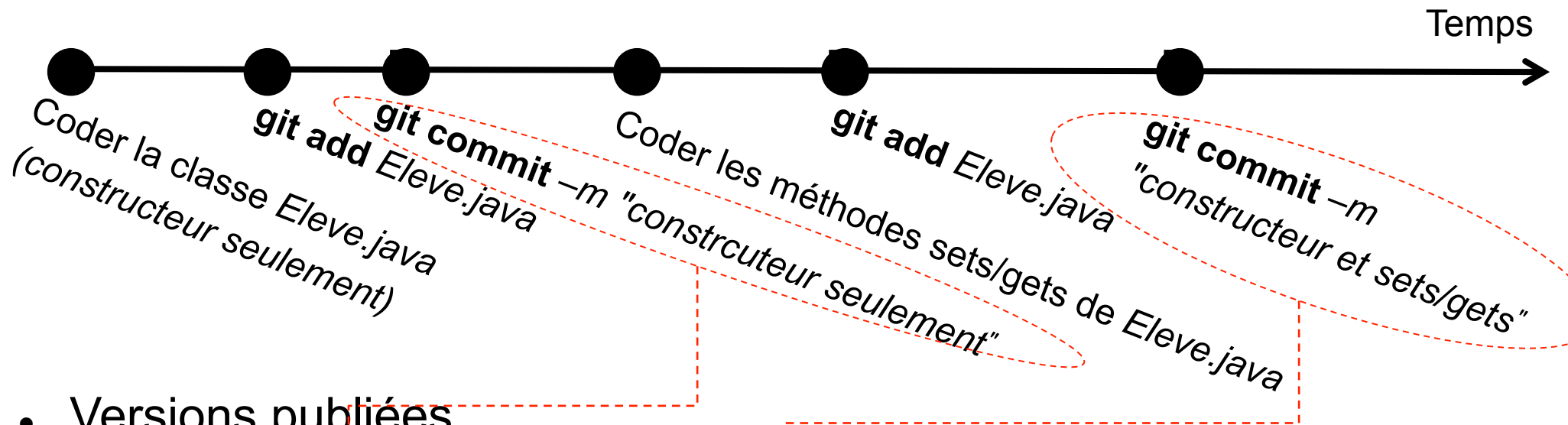
- Valider une version de code (passage du pré-dépôt au dépôt) :

\$ **git commit -m** *"première version : constructeur seulement"*

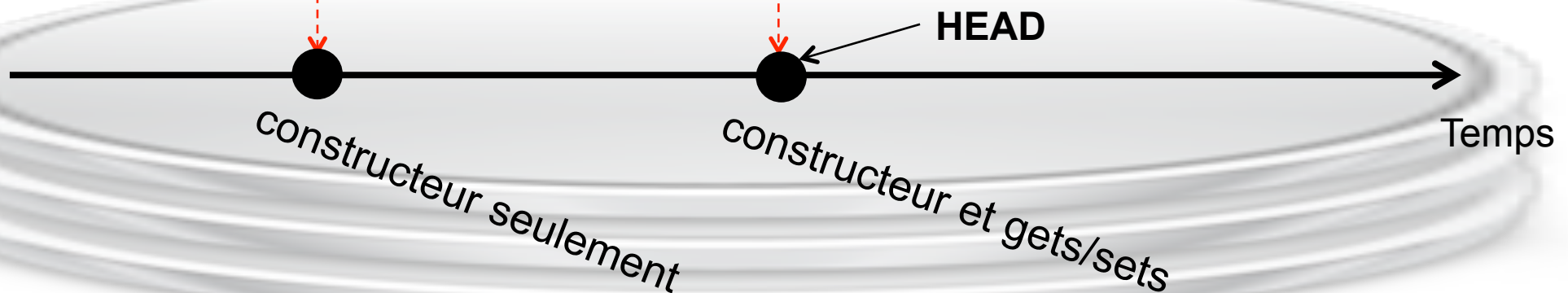


Git : validations ou dépôt des versions de code

- Développement de la classe *Eleve.java* et son versionning :



- Versions publiées



Dépôt local

Git : état des espaces

- Consulter l'état des fichiers de l'espace de travail et du pré-dépôt

\$ git status



```
MacBook-Pro-de-Mourad:Git ouziri$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        Eleve.java

nothing added to commit but untracked files present (use "git add" to track)
```

Git : état des espaces

- Consulter l'état des fichiers de l'espace de travail et du pré-dépôt

```
$ git add Eleve.java
```

```
$ git status
```



```
MacBook-Pro-de-Mourad:Git ouziri$ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to
       new file:   Eleve.java

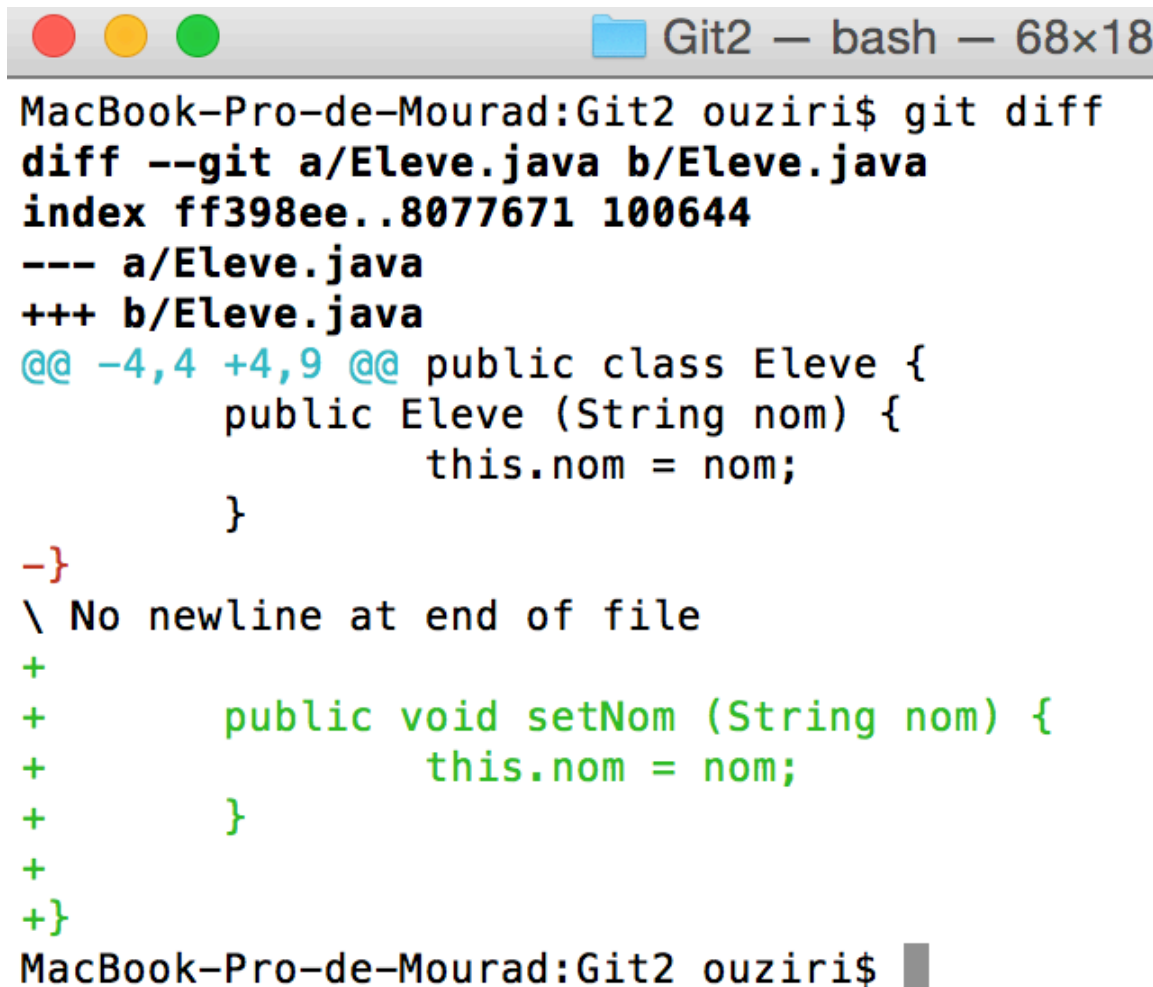
MacBook-Pro-de-Mourad:Git ouziri$ git status
On branch master

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)
       modified:   Eleve.java
```

Git : évolution de code

- Voir ce qui a changé : différence de code dans les trois espaces

\$ git diff

A screenshot of a macOS terminal window titled "Git2 — bash — 68x18". The terminal shows the command "git diff" being executed. The output displays the differences between the index and two working tree versions of the file "Eleve.java". The index version (ff398ee) has a class "Eleve" with a constructor. The working tree version "a/Eleve.java" is identical to the index. The working tree version "b/Eleve.java" has additional methods: "setNom" and a closing brace for the class.

```
MacBook-Pro-de-Mourad:Git2 ouziri$ git diff
diff --git a/Eleve.java b/Eleve.java
index ff398ee..8077671 100644
--- a/Eleve.java
+++ b/Eleve.java
@@ -4,4 +4,9 @@ public class Eleve {
     public Eleve (String nom) {
         this.nom = nom;
     }
-}
\ No newline at end of file
+
+     public void setNom (String nom) {
+         this.nom = nom;
+     }
+}
MacBook-Pro-de-Mourad:Git2 ouziri$
```

Git : évolution de code

- Voir ce qui a changé : différence de code entre...

\$ **git diff** : entre l'espace de travail et le pré-dépôt

-> exécuter \$ **git add** pour synchroniser les 2 espaces

\$ **git diff HEAD** : entre le répertoire de travail et la tête du dépôt

-> exécuter \$ **git add** suivi de \$ **git commit** pour synchroniser les 2 espaces

\$ **git commit -a** agrège les deux commandes **add** et **commit**

\$ **git diff --cached** : entre le pré-dépôt et la tête du dépôt

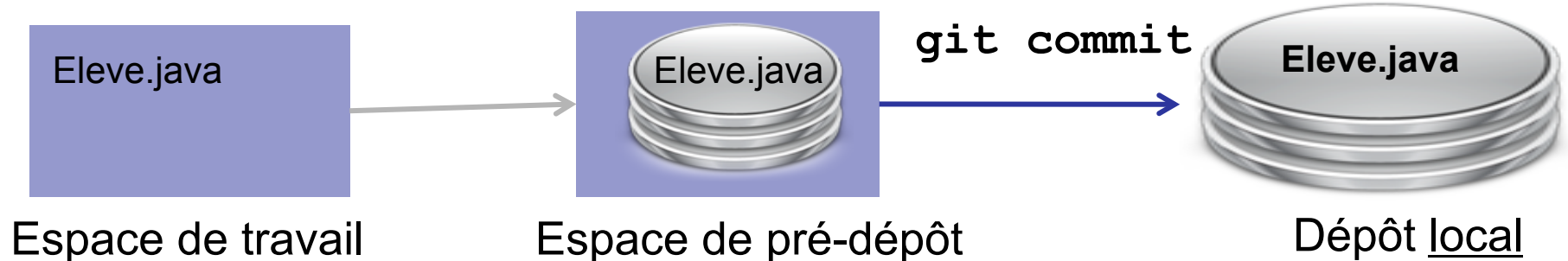
-> exécuter \$ **git commit** pour synchroniser

\$ **git diff --stat** : résumé des changements...

Git : validation d'une version

- Valider la version de code du pré-dépôt (la déposer dans le dépôt)

\$ **git commit** *Eleve.java -m "avec constructeur seulement"*

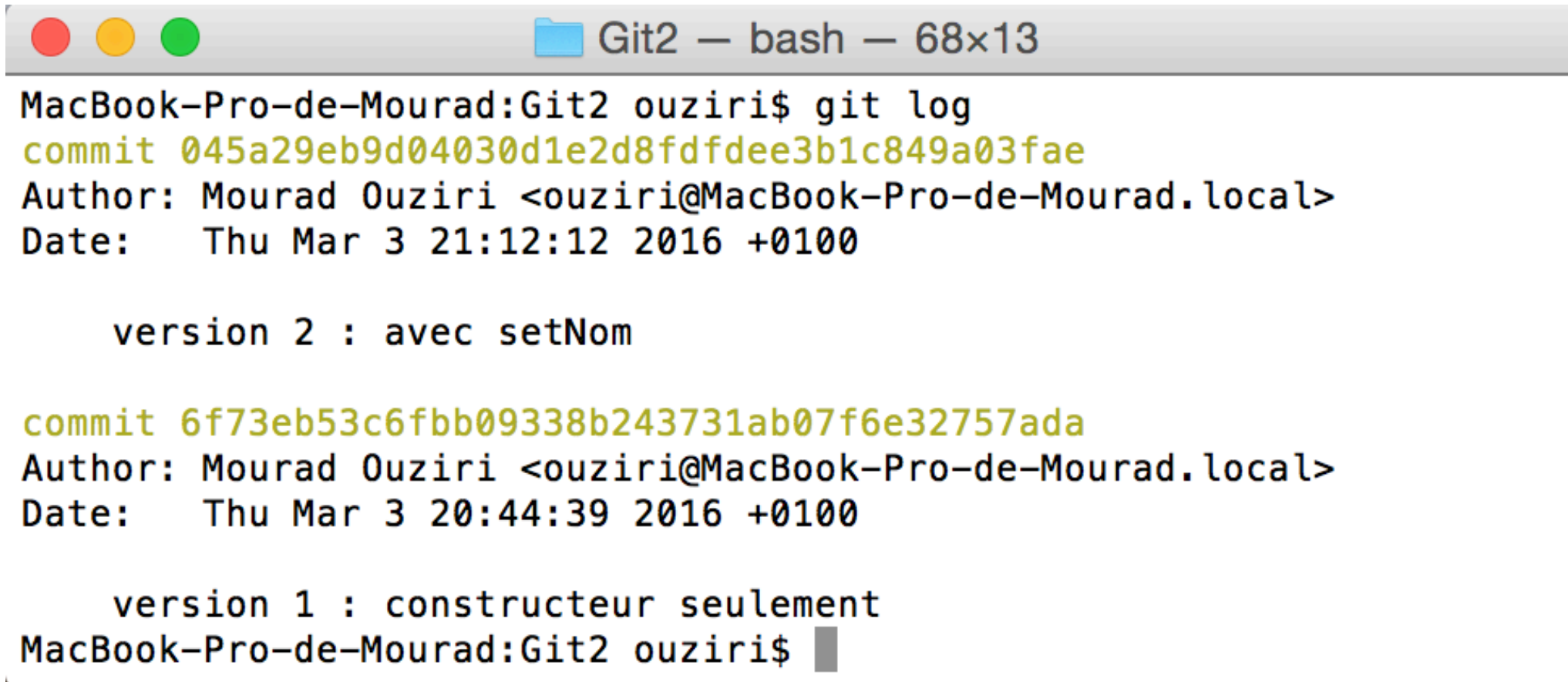


```
MacBook-Pro-de-Mourad:Git ouziri$ git status
On branch master
nothing to commit, working directory clean
MacBook-Pro-de-Mourad:Git ouziri$
MacBook-Pro-de-Mourad:Git ouziri$
```


Git : historique des versions

- Voir l'historique de validation des versions

\$ **git log** (options possibles : -p ou --stat)



```
MacBook-Pro-de-Mourad:Git2 ouziri$ git log
commit 045a29eb9d04030d1e2d8fdfdee3b1c849a03fae
Author: Mourad Ouziri <ouziri@MacBook-Pro-de-Mourad.local>
Date: Thu Mar 3 21:12:12 2016 +0100

    version 2 : avec setNom

commit 6f73eb53c6fbb09338b243731ab07f6e32757ada
Author: Mourad Ouziri <ouziri@MacBook-Pro-de-Mourad.local>
Date: Thu Mar 3 20:44:39 2016 +0100

    version 1 : constructeur seulement
MacBook-Pro-de-Mourad:Git2 ouziri$
```

Git : retour en arrière pour réparation

- Restaure l'espace de travail à partir du dépôt

\$ **git checkout** {HEAD, n°commit} **fichier.ext**

- Restaure l'espace de travail à partir du pré-dépôt

\$ **git checkout -- fichier**

- Reset current HEAD to the specified state

\$ **git reset** {--hard HEAD}

- Annule un commit dans le dépôt

\$ **git revert** {HEAD, HEAD^, n° du commit, ect.)

Git : manuel des commandes

- Manuel d'une commande

\$ **git help** commande

- Manuel des commandes Git

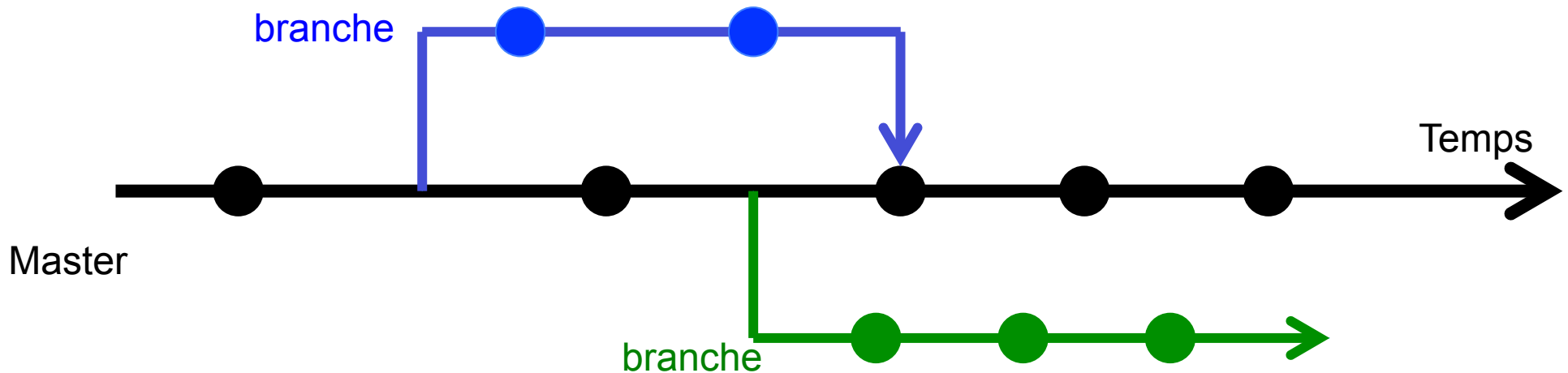
\$ **git help**

- Liste des commandes Git

\$ **git help -a**

Git : les branchements

- Créer un branche parallèle et annexe à l'axe principal (master) pour :
 - Développer de nouvelles versions sans altérer le développement principal
 - Lancer des développement expérimentaux ou exploratoires
 - Permettre à chacun (des développeurs) de mener des développements (créer un tracé de versions) de manière indépendante



Git : les branchements

- Créer une branche retourner l'objet Eleve au format html :

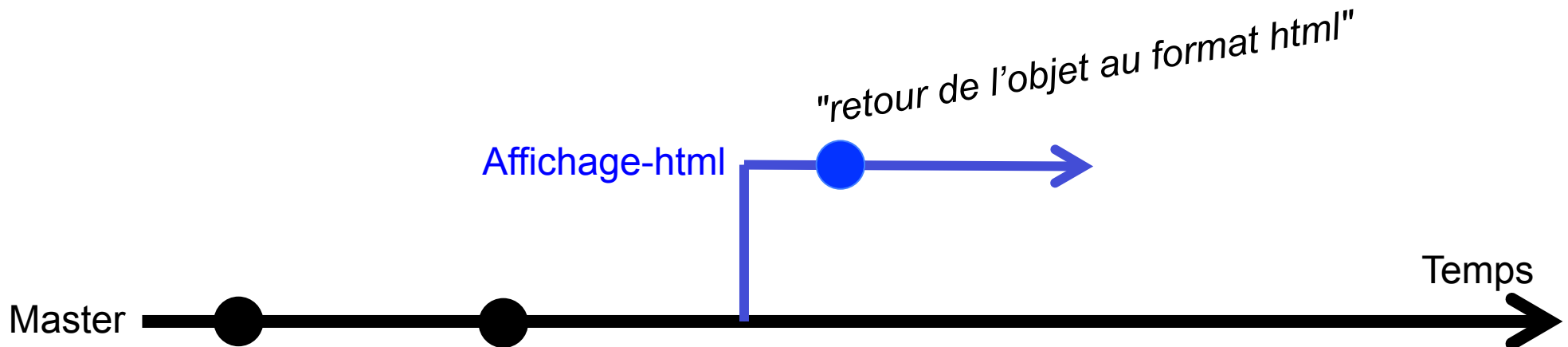
\$ git branch affichage-html

- Activer (se placer sur) la branche *affichage-html*

\$ git checkout affichage-html

// Développer la méthode getHTML() dans la classe Eleve sur la branche active, puis

\$ git add Eleve.java suivi de **\$ git commit --m "retour de l'objet au format html"**



Branchement : intégration d'une branche (merge)

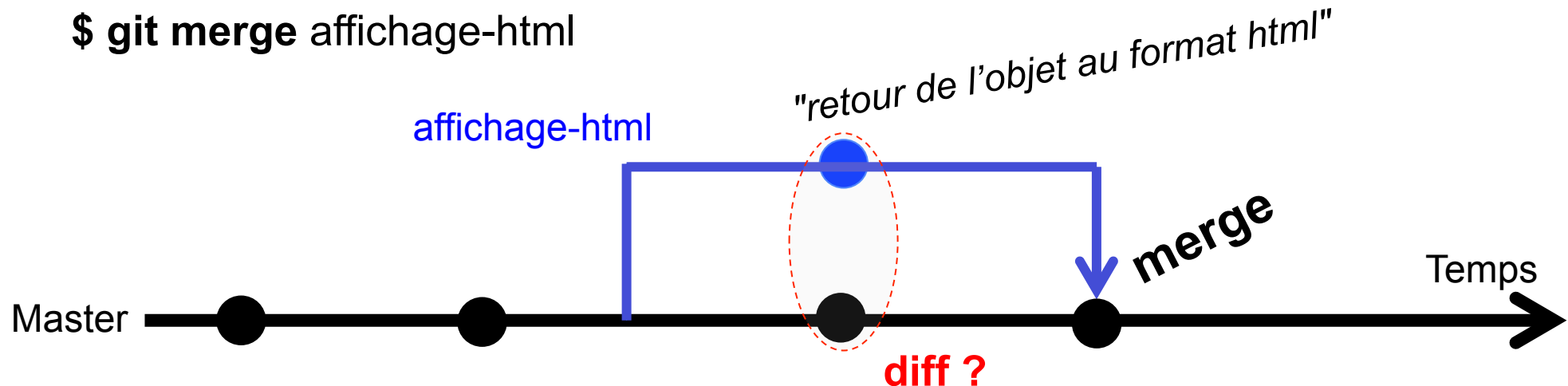
- Avant intégration d'une branche, on voudrait voir la différence entre le code de la branche et celui du master

```
$ git diff master..affichage-html
```

- Intégrer (prendre en compte) les versions développées sur une branche au master

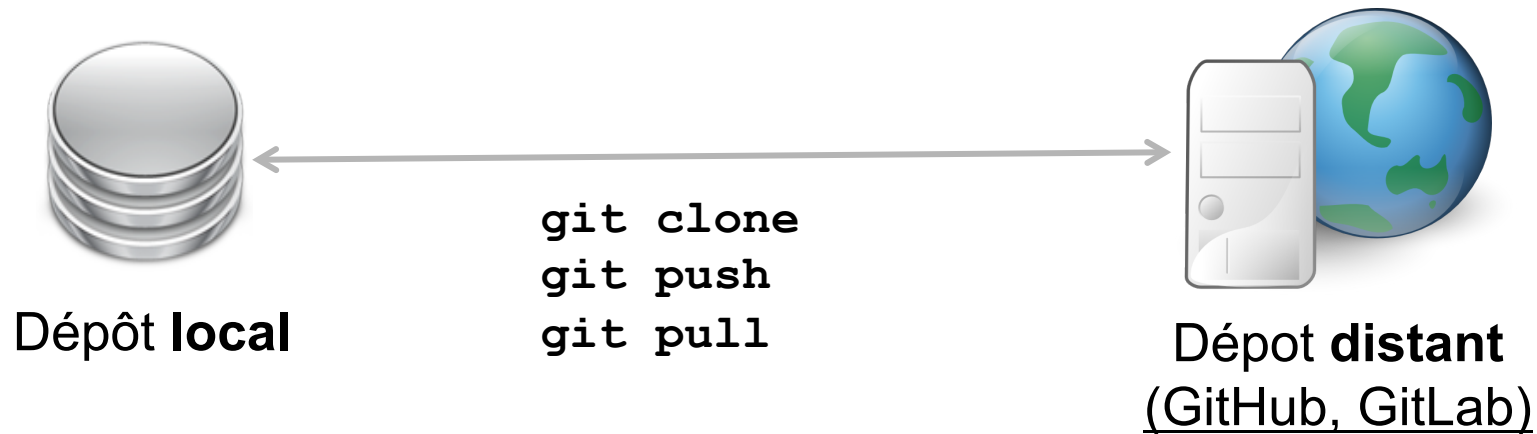
```
$ git checkout master // se mettre sur la branche principale
```

```
$ git merge affichage-html
```



Git en distribué : synchronisation avec des dépôts distants

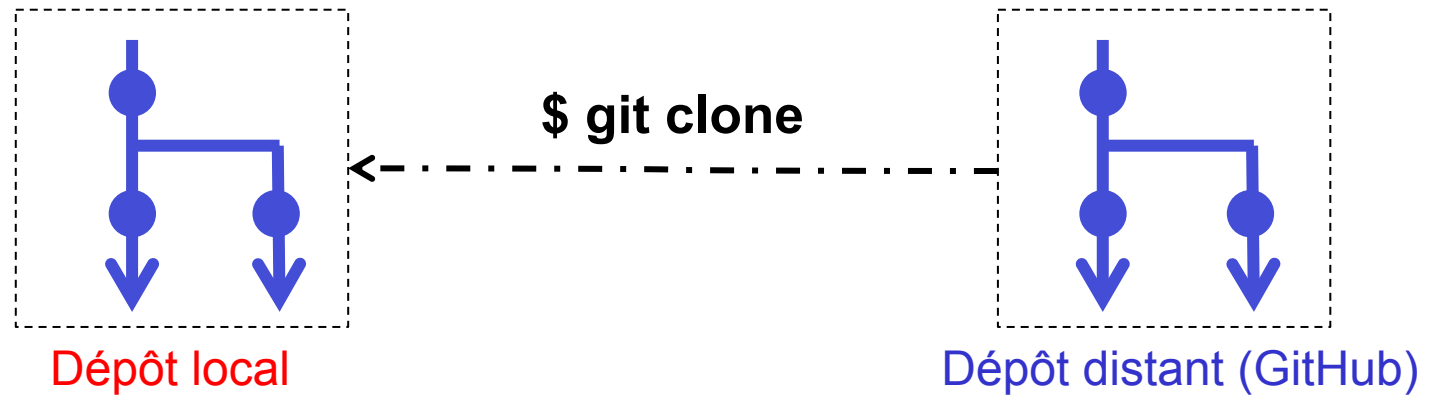
- Collaborer avec des développeurs distants



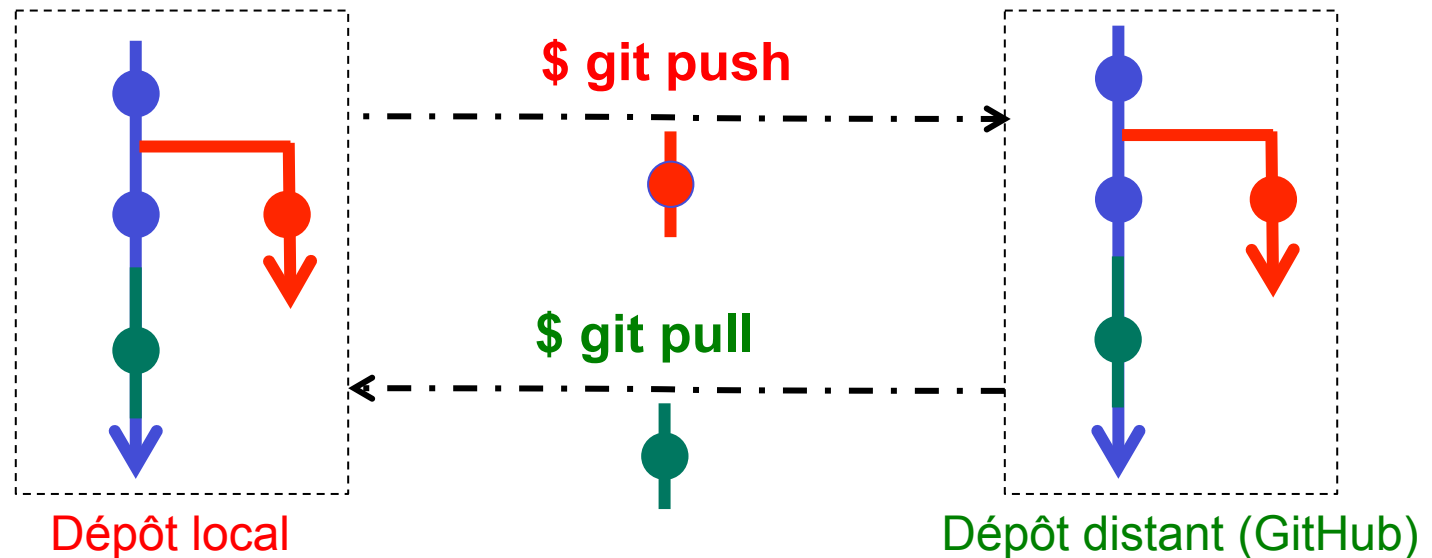
- Déposer ses versions (du dépôt local) sur un serveur commun (GitHub, GitLab)
- Récupérer les versions validées sur le serveur commun dans son dépôt local

Git en distribué : synchronisation avec des dépôts distants

- Initialiser/créer un dépôt local à partir d'un dépôt commun/distant :



- Pousser** son travail et **recupérer** celui partagé sur le dépôt distant :



Git en distribué : synchronisation avec des dépôts distants


- Créer un dépôt/projet sur GitHub/GitLab (hébergement de code)

<https://github.com/new>

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

 ouzirimo

Repository name

inscriptions-cours-bcp

Great repository names are short and memorable. Need inspiration? How about **potential-waddle**.

Description (optional)

Dépôt créé pour le cours de BCO



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

- Déclarer un dépôt distant

\$ **git remote add** <alias-depot-distant> <https://github.com/<username>/<projet>.git>

Pour voir les dépôts distants déclarés :

\$ **git remote -v**

Git en distribué : synchronisation avec des dépôts distants

- Cloner un dépôt/projet distant :

\$ **git clone** <url-dépôt>

Exemple : \$ **git clone** <https://github.com/ouzirimo/dev-cours-bco.git>

- Déposer (publier ou partager) son dépôt local sur serveur d'hébergement partagé

\$ **git push** <nom-depot-distant> <branche-locale>

Exemple : \$ **git push** projet2-github master

- Récupérer une branche de versions déposées sur le dépôt distant dans son dépôt local (pull = fetch + merge) :

\$ **git pull** <nom-depot-distant> <branche-locale>

Exemple : \$ **git pull** projet2-github master

Git en version graphique : **GitHub Desktop App**

GitHub Desktop

[Overview](#) | [Release Notes](#) | [Help](#)

Simple collaboration from your desktop

GitHub Desktop is a seamless way to contribute to projects on **GitHub** and **GitHub Enterprise**.

Available for [Mac](#) and **Windows**

Download GitHub Desktop

OS X 10.9 or later

By clicking the Download button you agree to the **End-User License Agreement**

