Système 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

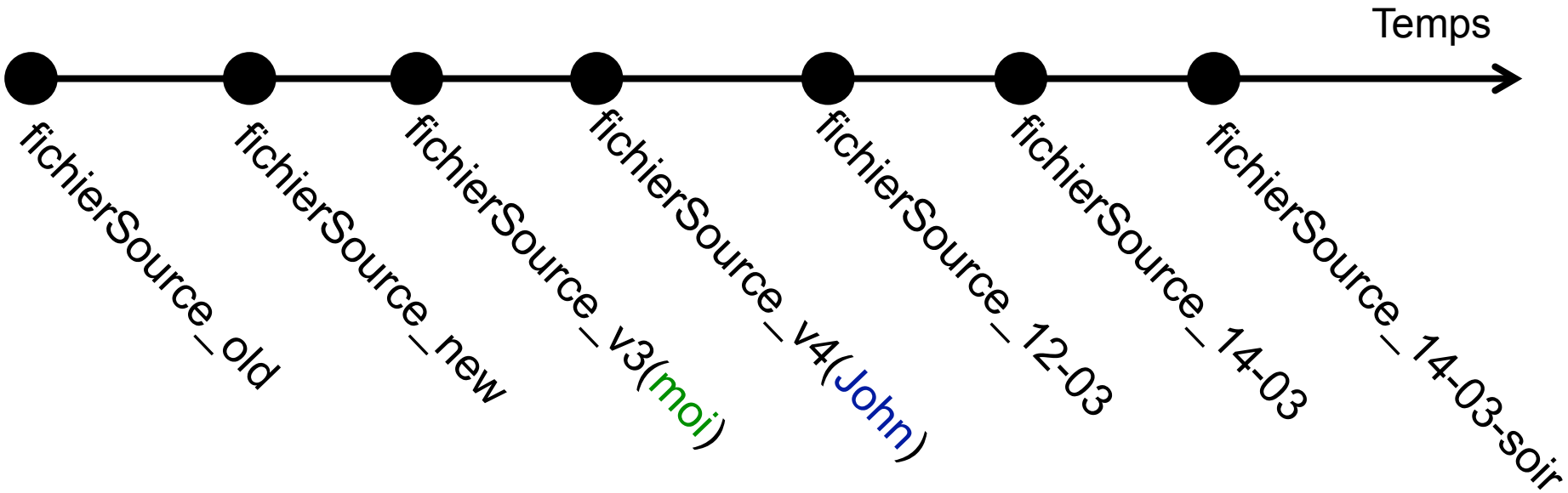
IUT de Paris Descartes

PARIS DESCARTES IUT

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

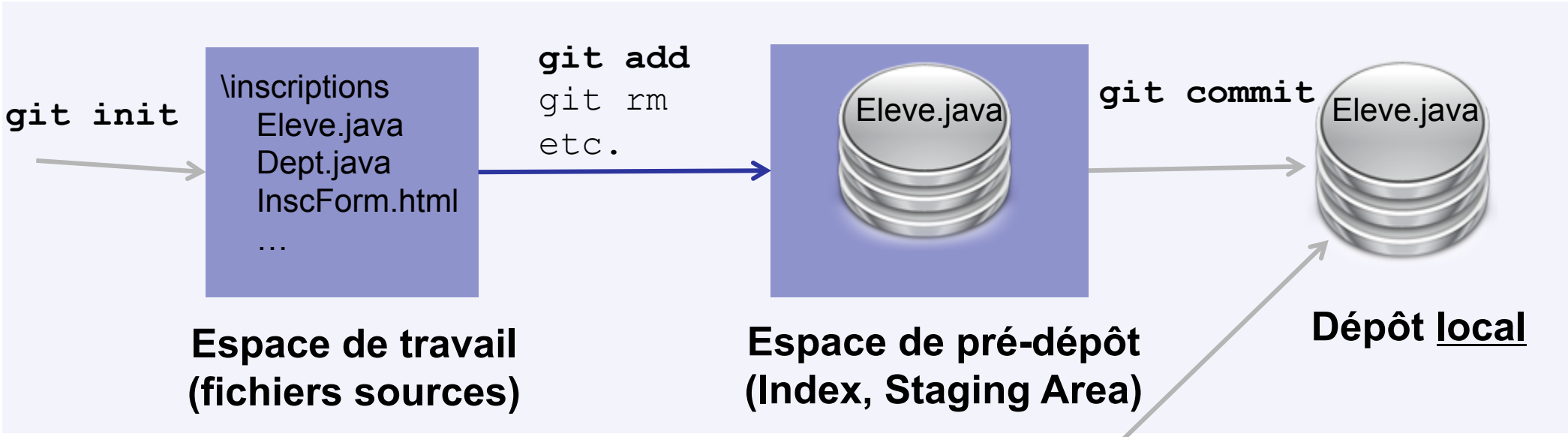
GIT : un outil de suivi de versions de code

- 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 clone
git push
git pull
```

Dépôt distant
(GitHub, GitLab)

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
- **Index ou espace de pré-dépôt (staging area)** : espace intermédiaire où sont déposées les versions de avant publication/ validation (commit) et mise en dépôt

Git : initalisation d'un projet/dépôt

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

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

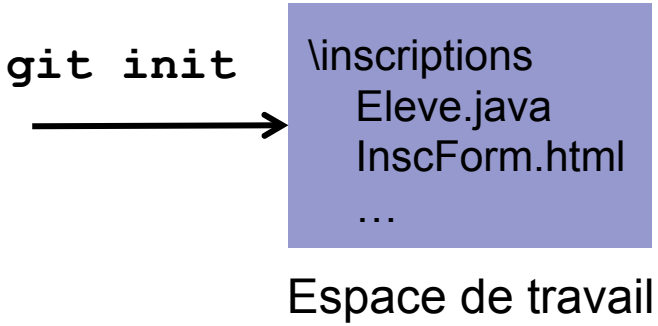
  $ git config --global user.email mourad.ouziri@parisdescartes.fr

  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

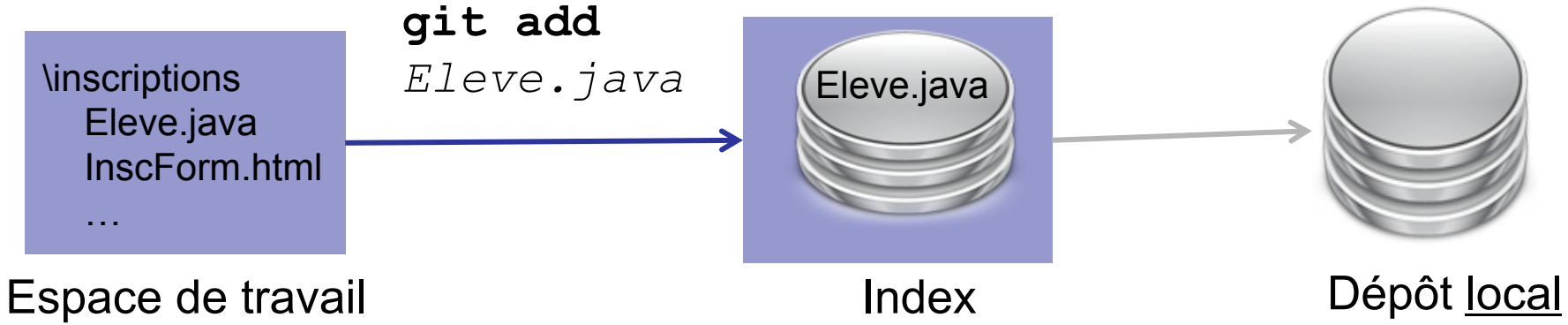
  $ **git init**



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

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

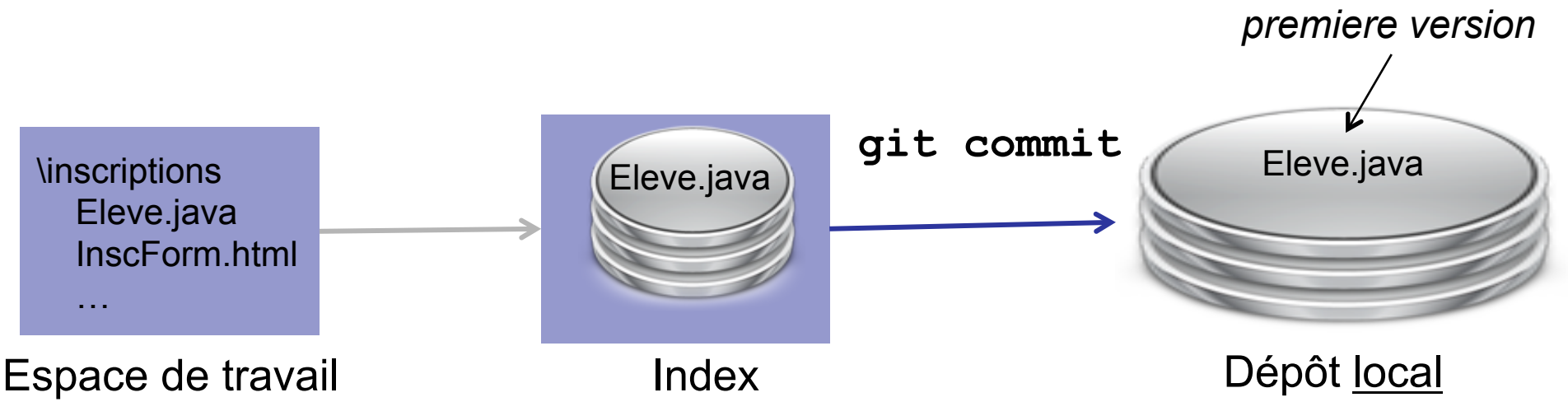
  $ **git add** *Eleve.java*



Git : validation (commit) d'une version

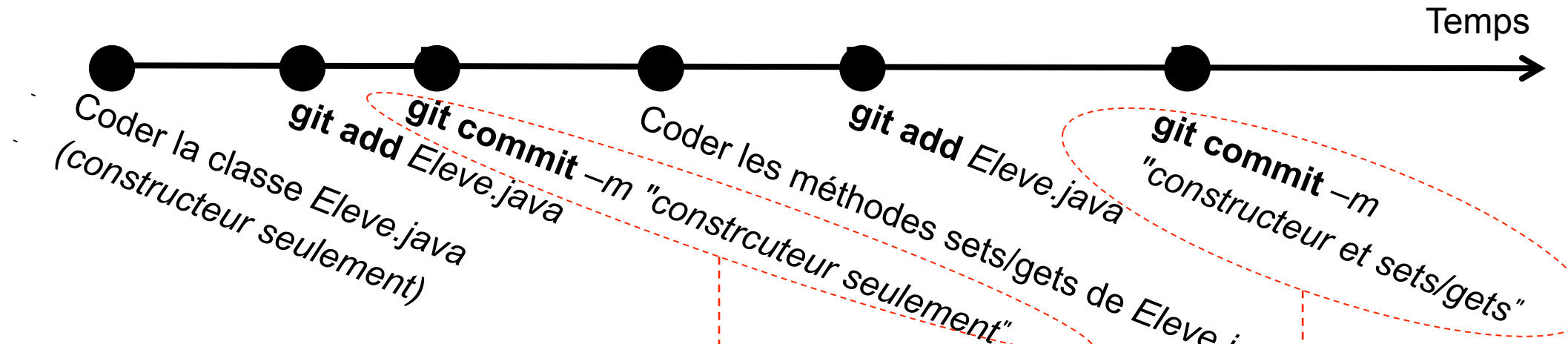
- Valider une version de code (passage du pré-dépôt au dépôt) :

  $ **git commit** *–m "première version : constructeur seulement"*

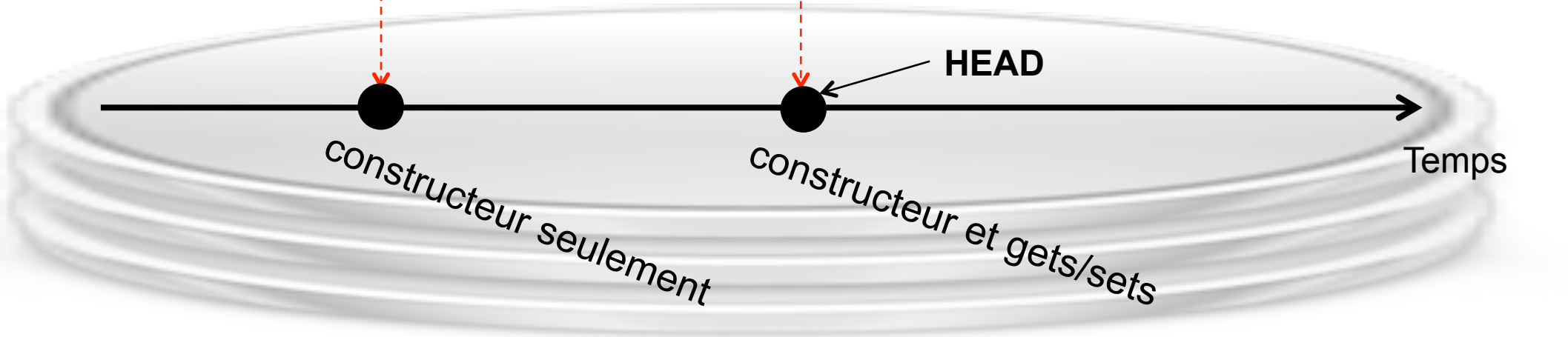


Git : validation ou dépôt des versions de code

- 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

Espace de travail | Index | Dépôt local

```
Git — bash — 80×24
MacBook-Pro-de-Mourad:Git ouziri$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to includ

        Eleve.java

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

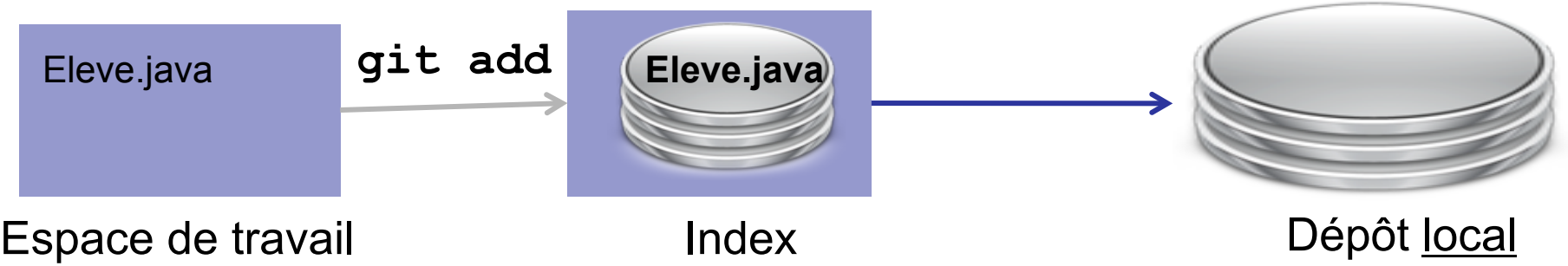
```
MacBook-Pro-de-Mourad:Git ouziri$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will b
  (use "git checkout -- <file>..." to discard ch

        modified:   Eleve.java

no changes added to commit (use "git add" and/or
```

Git : état des espaces

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

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

  $ **git status**



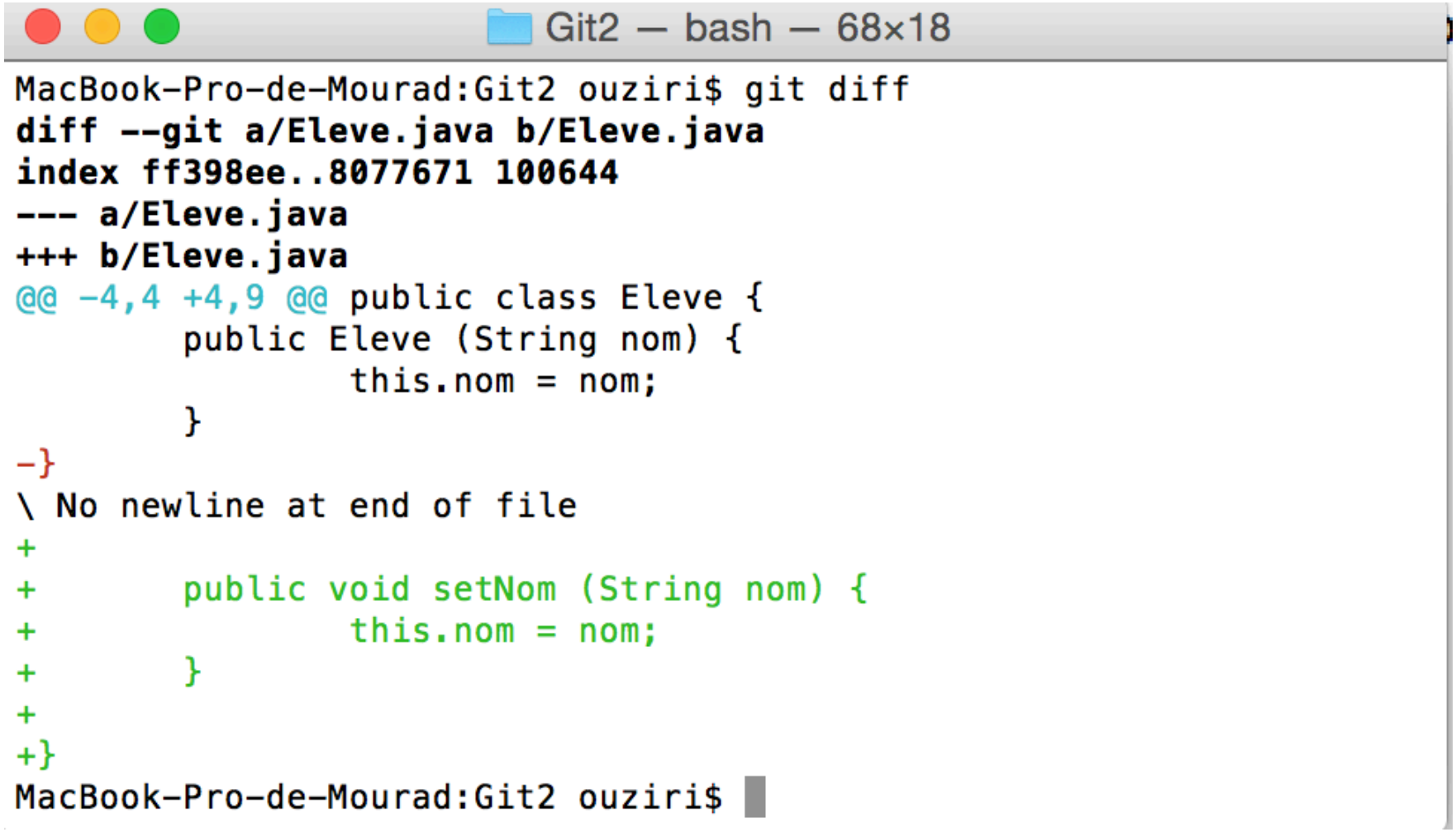

```
Git — bash — 80×13
MacBook-Pro-de-Mourad:Git ouziri$ git status
On branch master

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**



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

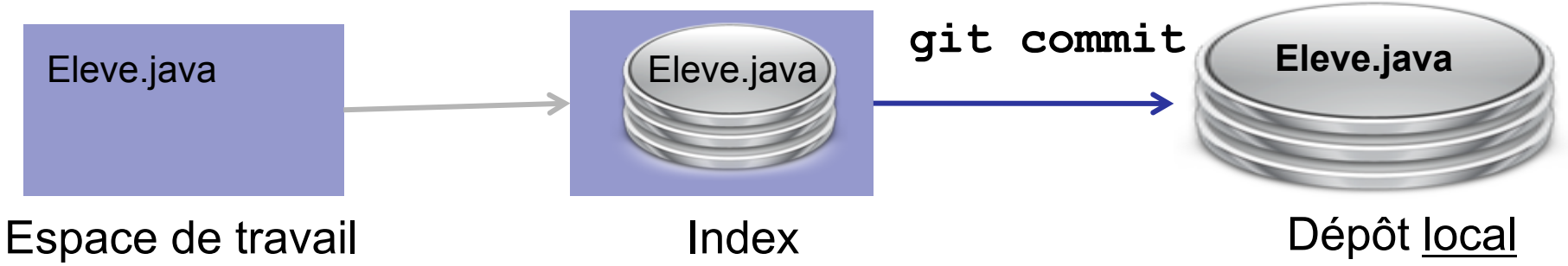
  -> exécuter **$ git commit** pour synchroniser

  $ **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"*

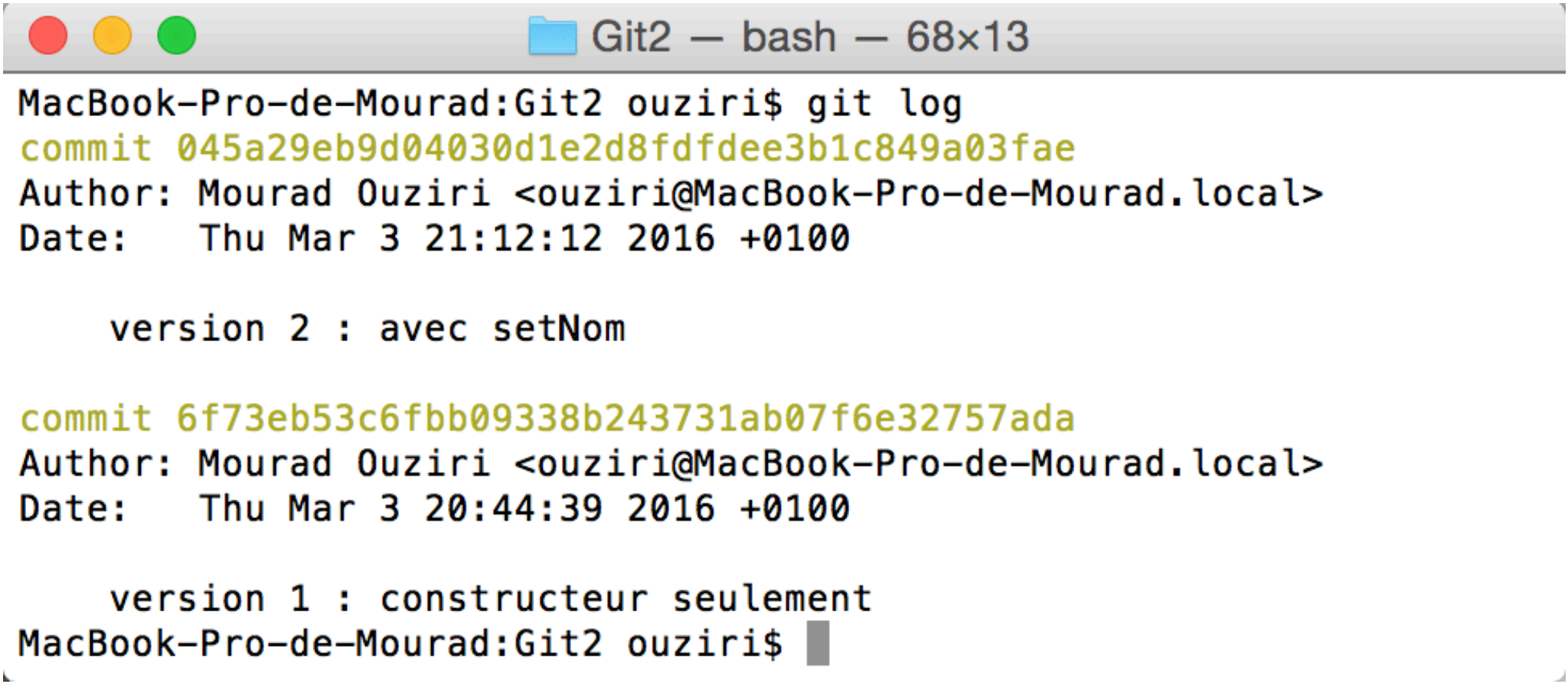



```
Git — bash — 82×5
MacBook-Pro-de-Mourad:Git ouziri$ git status
On branch master
nothing to commit, working directory clean
MacBook-Pro-de-Mourad:Git ouziri$
MacBook-Pro-de-Mourad:Git ouziri$
```

Git : historique des versions

- Voir l'historique de validation des versions

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



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

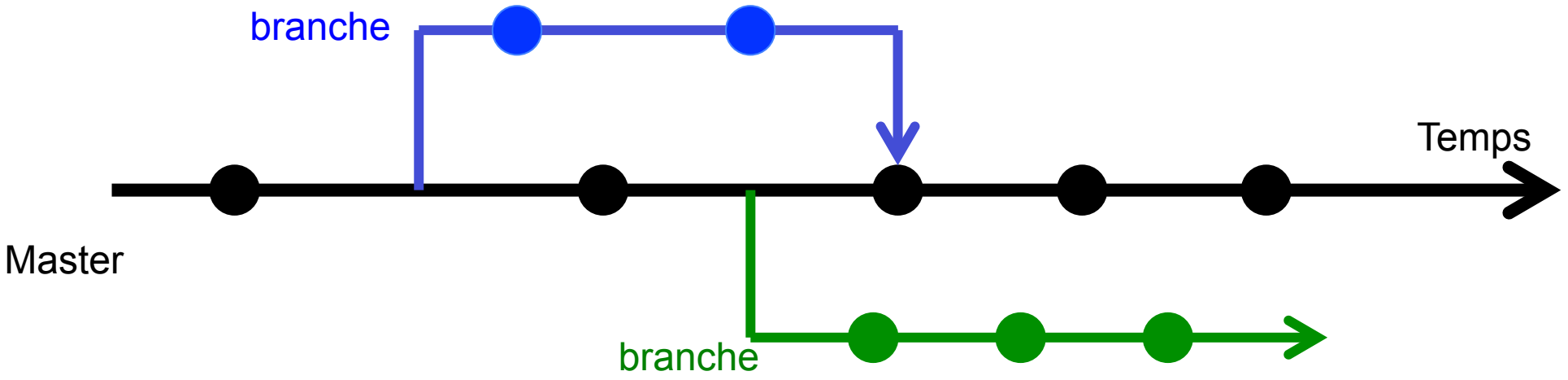
  $ **git help**

- 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 :

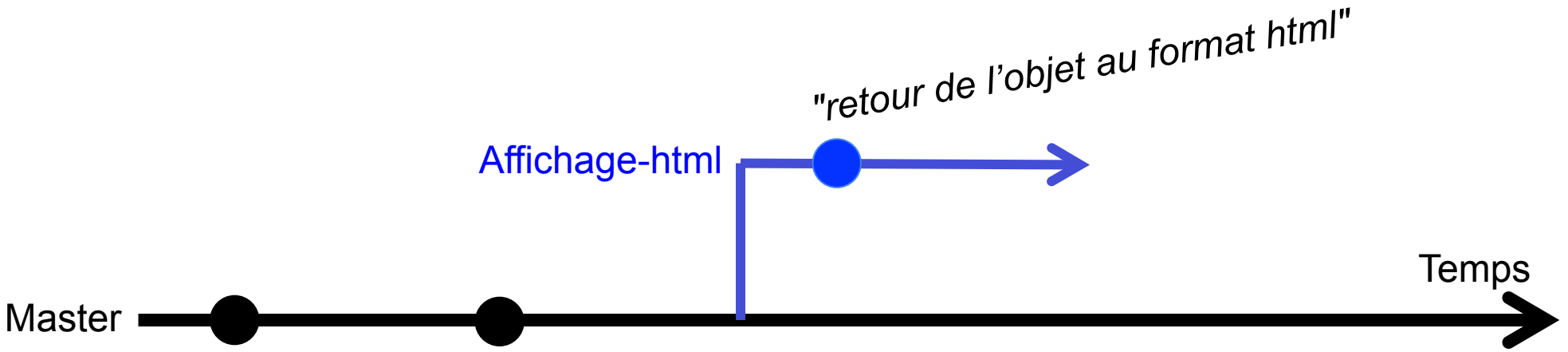
  $ git branch affichage-html

- 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)

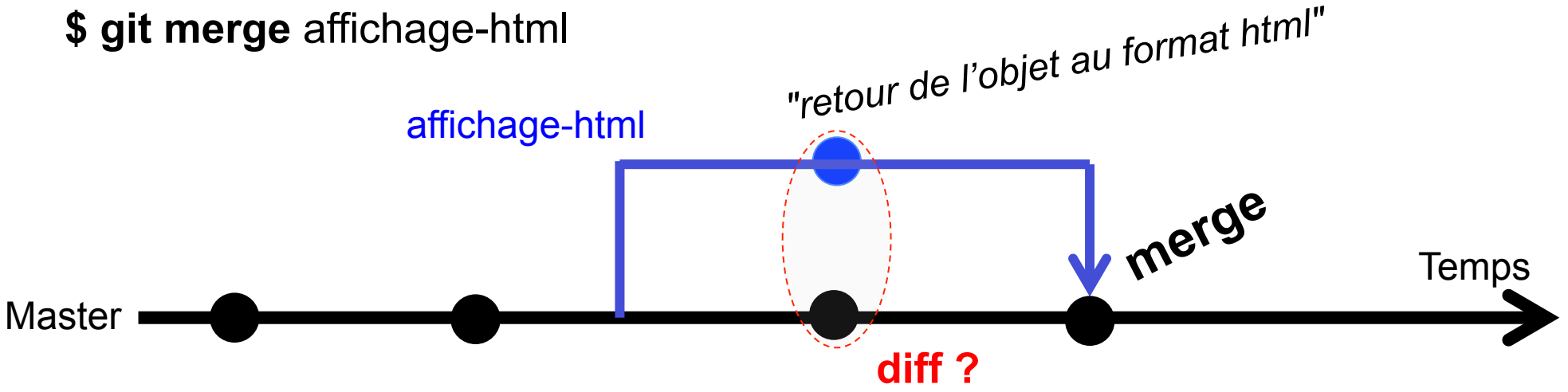
- Avant intégration d'une branche, on voudrait voir la différence entre le code de la branhce et celui du master

  $ git diff master..affichage-html

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

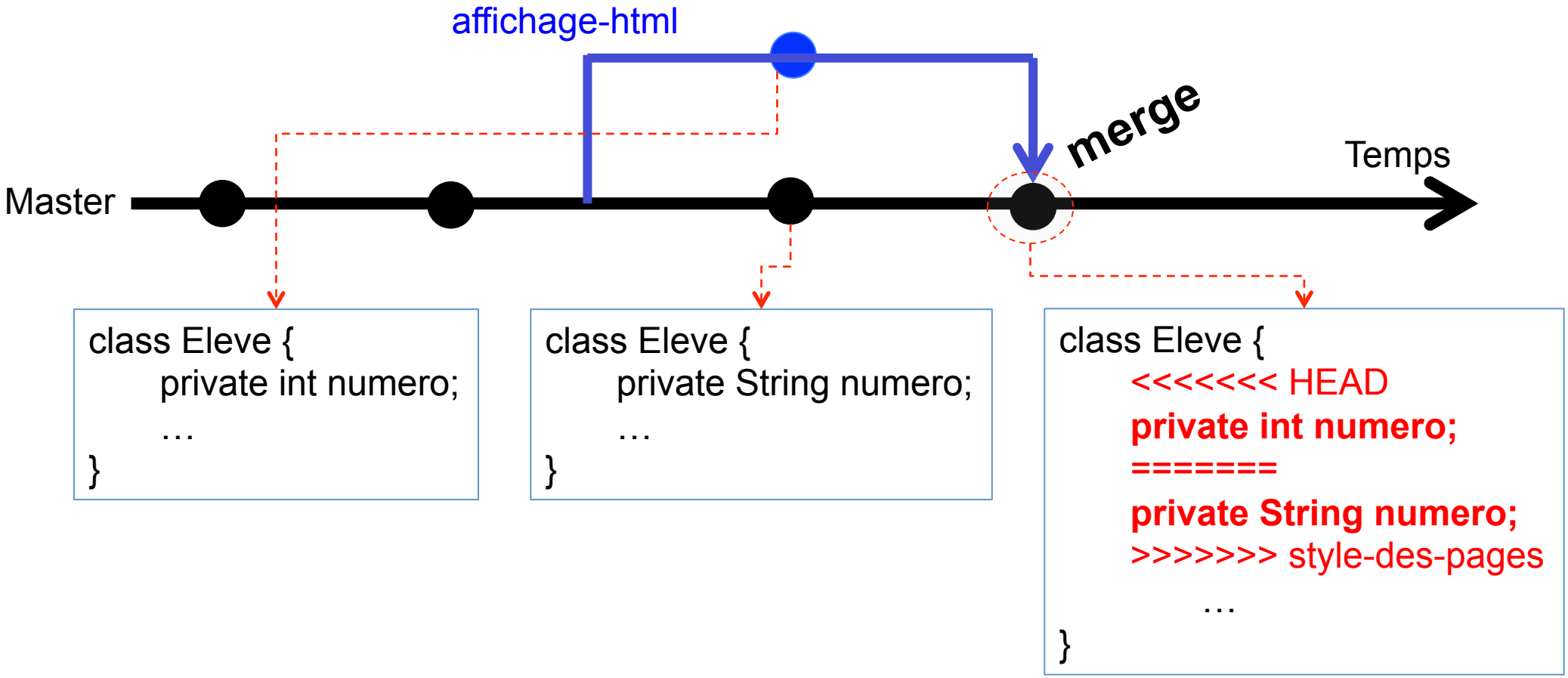
  $ git checkout master // se mettre sur la branche principale

  $ git merge affichage-html



Branchement : gestion de conflits

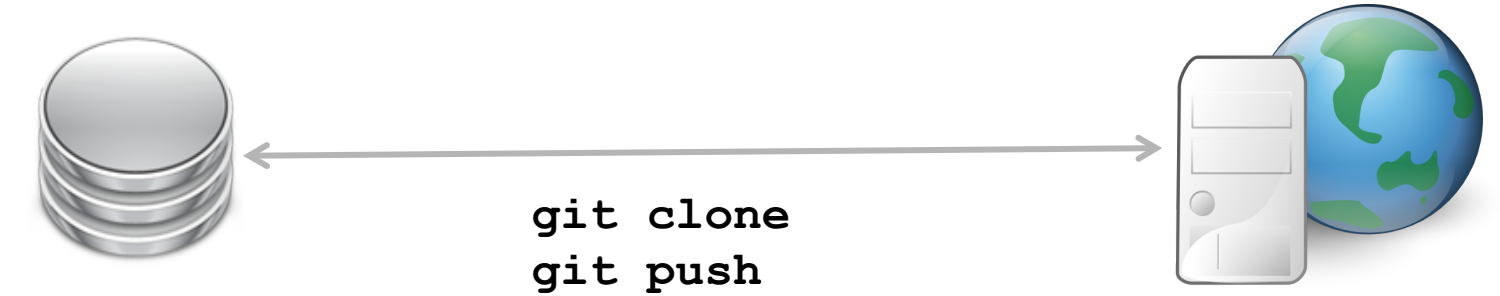
- Lors de l' intégration d'une branche, des conflits pourraient être levés par Git, notamment si la même ligne est modifiée différemment



- Le lignes en conflit est marqué: le résoudre manuellement !

Git en distribué : synchronisation avec des dépôts distants

- Collaborer avec des développeurs distants



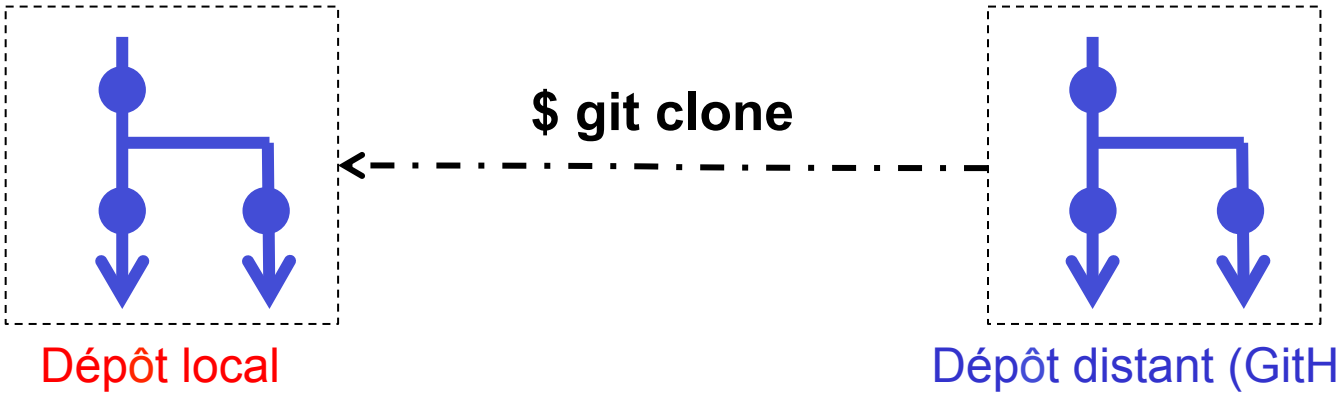
Dépôt **local**

Dépot **distant**
(GitHub, GitLab)

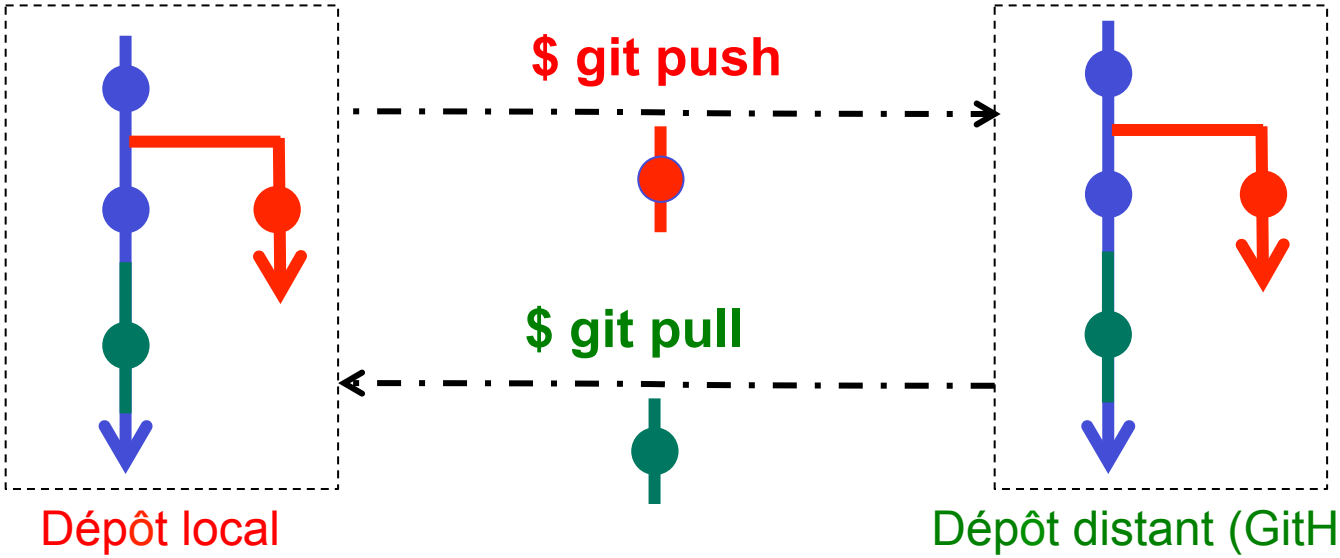
- Déposer ses versions (du dépôt local) sur un serveur commun (GitHub, GitLab)
- Récupérer les versions validées sur le serveur commun dans son dépôt local

Git en distribué : synchronisation avec des dépôts distants

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



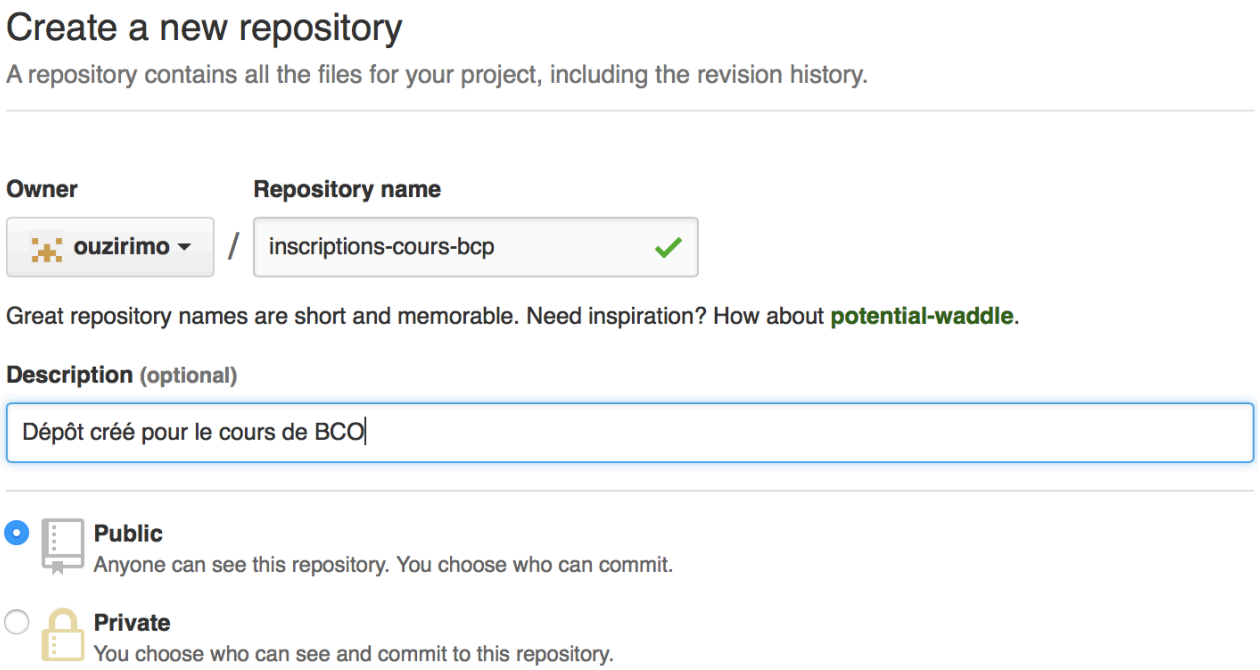
- Pousser son travail et récupérer celui partagé sur le dépôt distant :



Git en distribué : synchronisation avec des dépôts distants

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

https://github.com/new



- 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**