ULB

UNIVERSITÉ LIBRE DE BRUXELLES

INFO-F-307 - Séance 1
Git (Basics and Branching)

Axel Abels - aabels@ulb.ac.be
Jacopo De Stefani - jdestefa@ulb.ac.be
Jerôme De Boeck - jdeboeck@ulb.ac.be
Nassim Versbraegen - nversbra@ulb.ac.be

Université Libre de Bruxelles
A.A. 2018-2019

5 février 2019

ULB

UNIVERSITÉ LIBRE DE BRUXELLES

Introduction

Git Basics

Git branching

ULB

UNIVERSITÉ LIBRE DE BRUXELLES

Introduction

Les assistants

Axel Abels

- E-mail : aabels@ulb.ac.be
- Bureau : 2.N.8.204

Jacopo De Stefani

- E-mail : jdestefa@ulb.ac.be
- Bureau : 2.08.212

Jerôme De Boeck

- E-mail : jdeboeck@ulb.ac.be
- Bureau : 2.N.3.207

Nassim Versbraegen

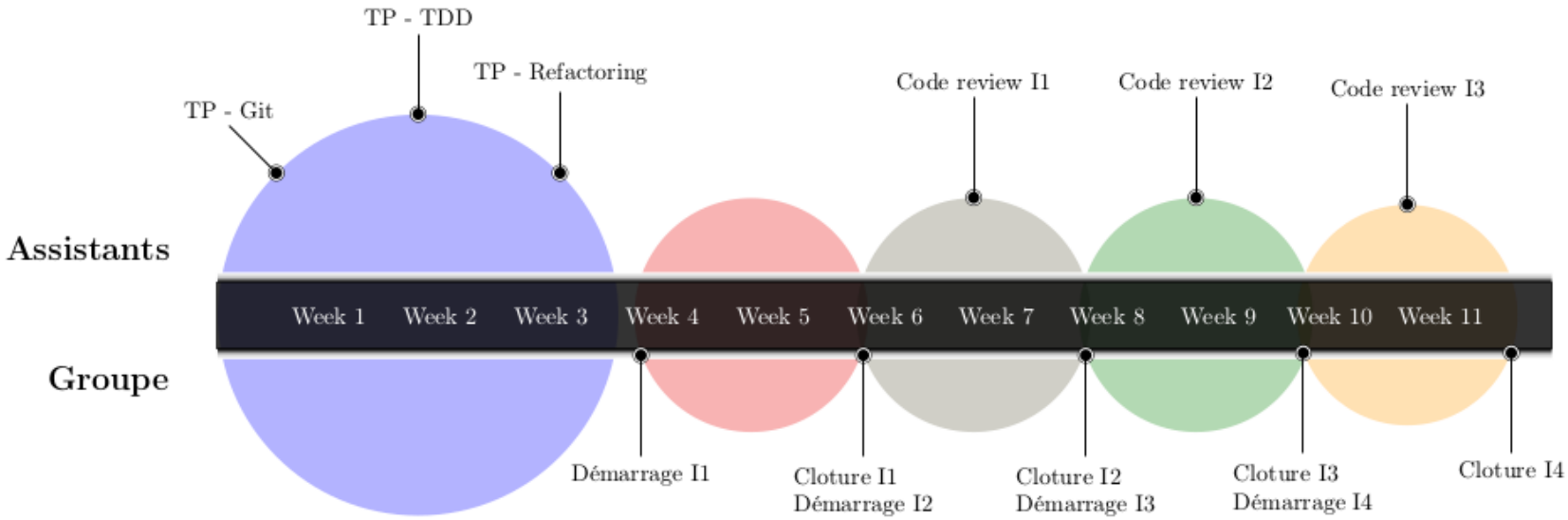
- E-mail : nversbra@ulb.ac.be
- Bureau : 2.08.213

Pour toute communication concernant le projet :
infof307@gmail.com

Projet (1)

- Application avec GUI en Java.
- Groupes de 8 personnes.
- Approche eXtreme Programming (XP) - 4 itérations.
- Pour chaque itération :
  - Démo
  - Fiches itération
  - Code itération (Gitlab)
- Examen = Présentation du projet (Itération 4)
- Taille $\text{projet}_{\text{Sciences}} \neq \text{Taille projet}_{\text{Polytech/CyberSec}}$
  - Cf. Slides Introduction

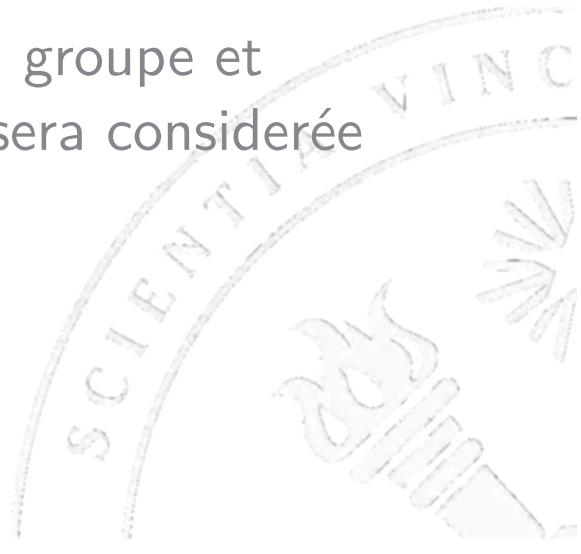
Projet (2)


TP - TDD
TP - Refactoring
TP - Git
Code review I1
Code review I2
Code review I3
Assistants
Groupe
Week 1
Week 2
Week 3
Week 4
Week 5
Week 6
Week 7
Week 8
Week 9
Week 10
Week 11
Démarrage I1
Cloture I1
Démarrage I2
Cloture I2
Démarrage I3
Cloture I3
Démarrage I4
Cloture I4

Projet (3)

TO DO - Avant 15/02 :

- Créer une compte Gitlab avec le NetID ULB comme nom d'utilisateur.
- Envoyer la composition et les noms utilisateur Gitlab pour les groupes du projet à `infof307@gmail.com`.
- Si vous n'arrivez pas à former un groupe, envoyez quand même un mail pour prévenir les assistants.
- Toute personne ne faisant pas partie d'aucun groupe et n'ayant pas communiqué avec les assistants sera considerée comme exclue du projet $\Rightarrow$ Note finale nulle



ULB

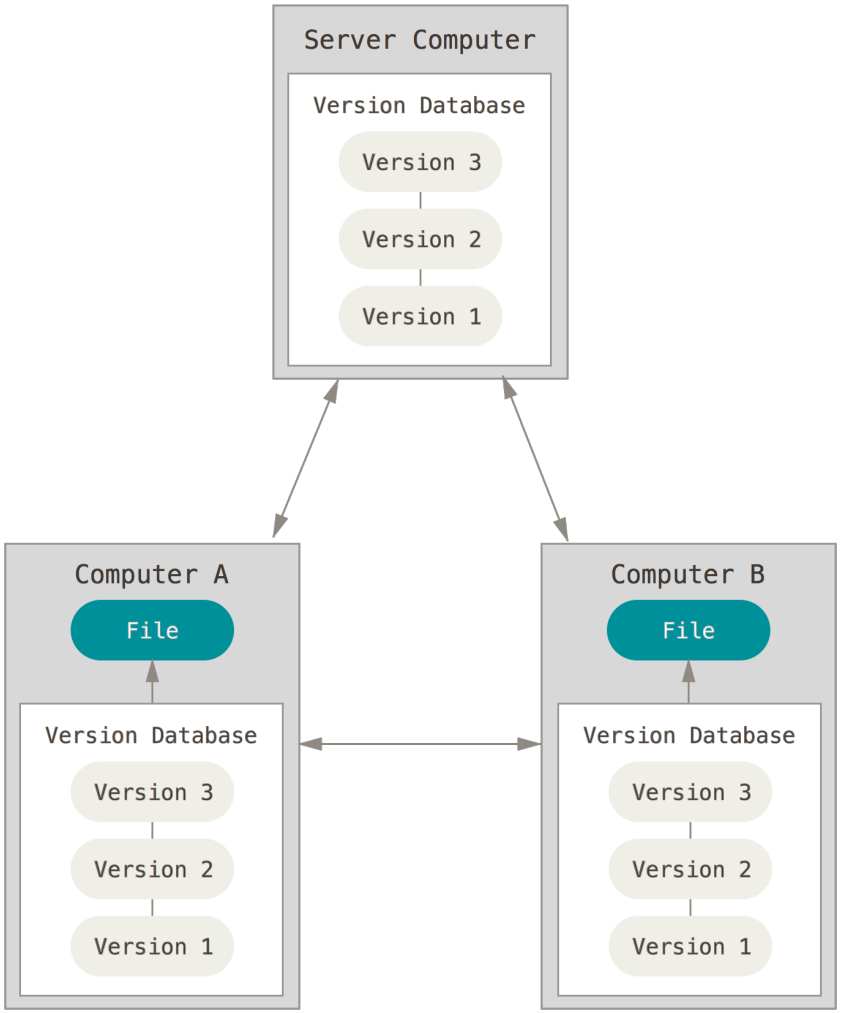
UNIVERSITÉ LIBRE DE BRUXELLES

Git Basics

Git ?

Relevant XCKD : 1597

Git : En réalité



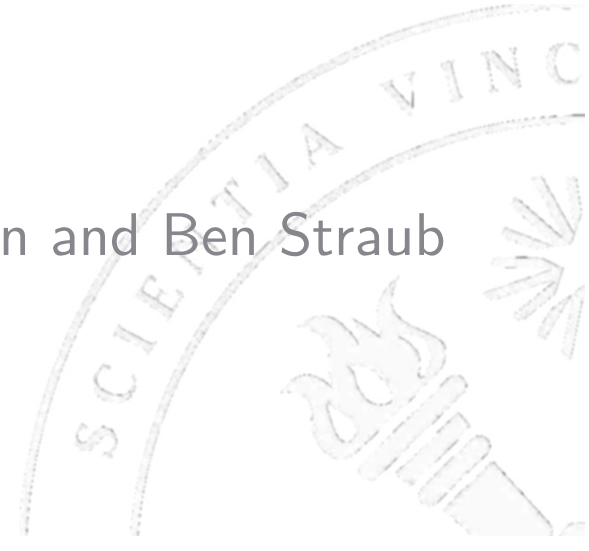
Source : Pro Git book by Scott Chacon and Ben Straub

Initialisation d'un depôt

```
$ git init
```

- Crée un répertoire `.git` dans le répertoire courant qui contient tous les metadonnées nécessaires au dépôt.
- Aucun fichier est sous contrôle de version.

Source : Pro Git book by Scott Chacon and Ben Straub

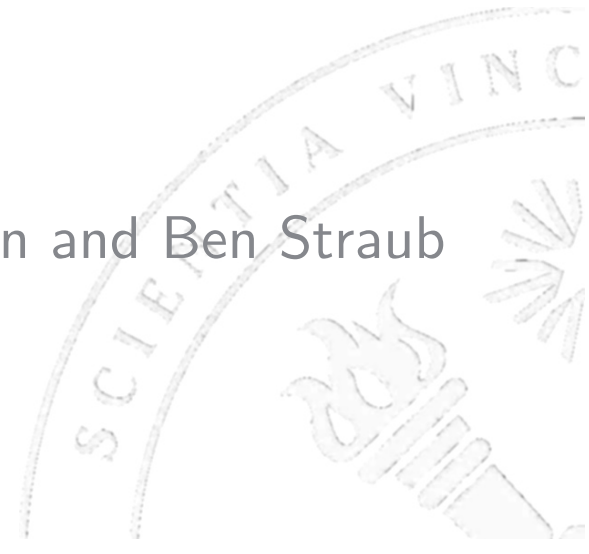


Clonage d'un depôt

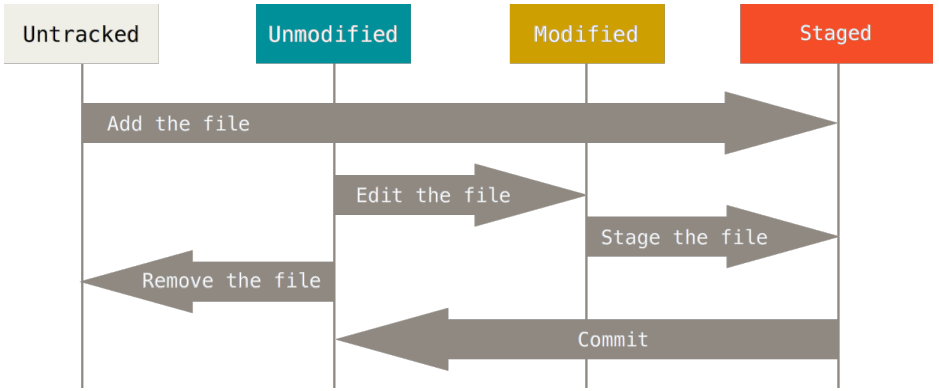
```
$ git clone [url]
```

- Crée une copie locale du depôt Git stocké à l'adresse `clone`.
- La copie est une image du depôt au moment où le clonage a été effectué.

Source : Pro Git book by Scott Chacon and Ben Straub



Cycle de vie d'un fichier



- `$ git add [file]`
  Ajoute `[file]` au système du suivi de version.
- `$ git rm [file]`
  Retire `[file]` du système du suivi de version et supprime le fichier correspondant dans la copie locale.
- `$ git rm --cached`
  Retire `[file]` du système du suivi de version.

Source : Pro Git book by Scott Chacon and Ben Straub

Commit

```
$ git commit
```

- Valide les modifications (commit) effectuées sur la copie locale du dépôt et enregistre le commit dans les metadonnées du dépot.

Adaptée de : Pro Git book by Scott Chacon and Ben Straub

Relevant XKCD : 1296

Afficher l'état d'un dépôt Git

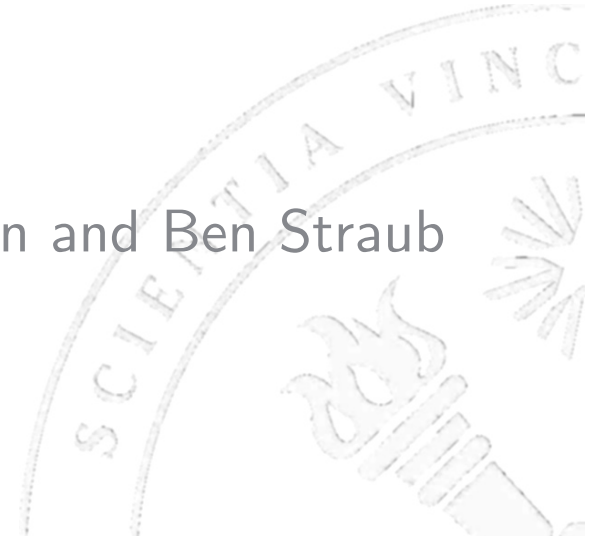
- `$ git status`
  Affiche l'état des fichiers sous suivi de version.
- `$ git log`
  Visualise l'histoire des commits.

Adaptée de : Pro Git book by Scott Chacon and Ben Straub

Travailler avec des dépôts distants (1)

- `$ git remote -v`
  Affiche les dépôts distants couramment référencés.
- `$ git remote add [remote-name] [url]`
  Ajoute une référence nommée `[remote-name]` au dépôt distant situé sur `[url]`.
- `$ git remote rm [remote-name]`
  Supprime la référence nommée `[remote-name]`.

Adaptée de : Pro Git book by Scott Chacon and Ben Straub



Travailler avec des dépôts distants (2)

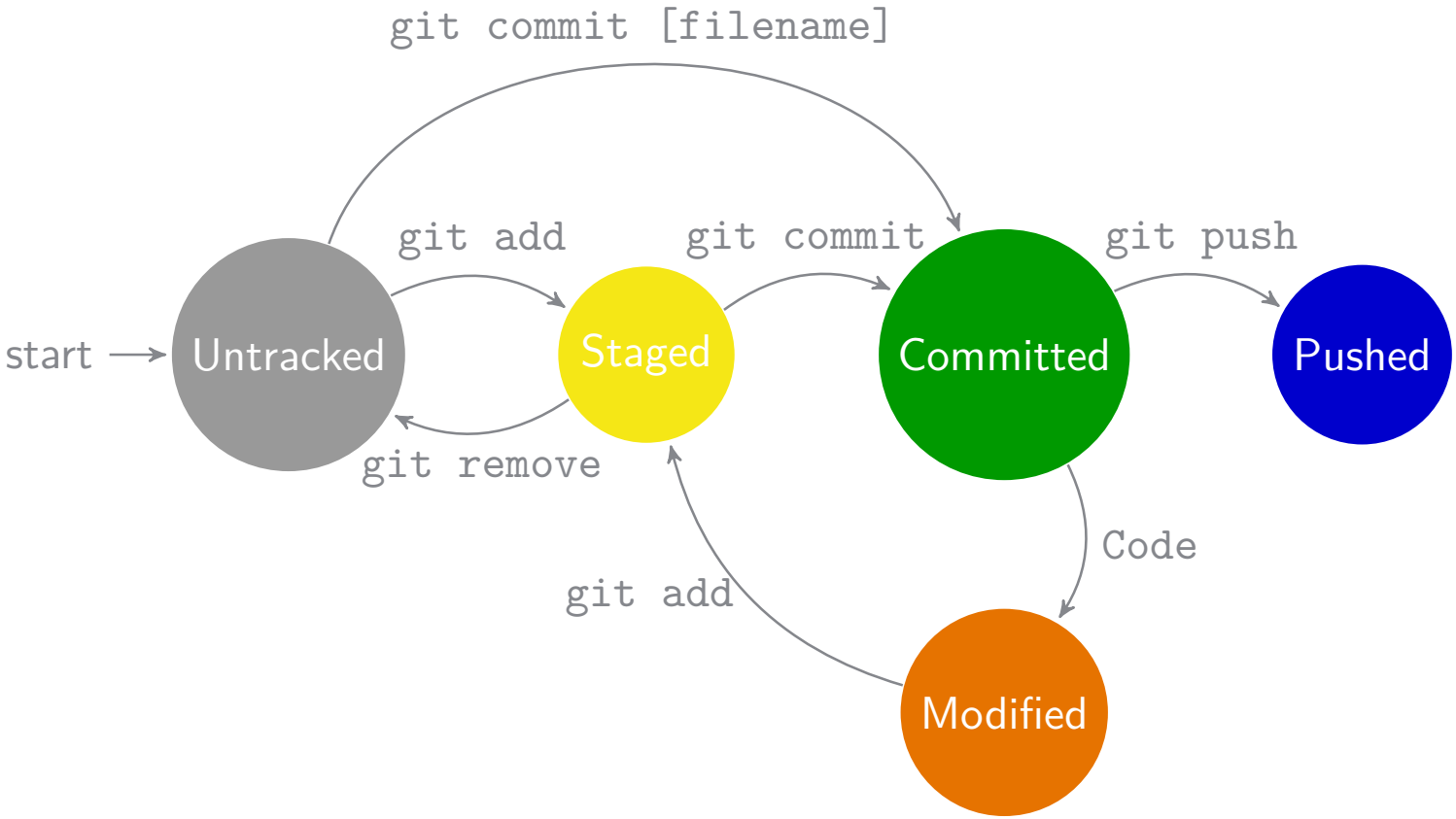
- `$ git fetch [remote-name]`
  Récupère toutes les données présent sur le dépot distant qui ne sont pas présents dans le dépot local.
- `$ git pull`
  Récupère et fusionne automatiquement une branche distante dans votre branche locale.
- `$ git push [remote-name] [branch-name]`
  Pousse la branche `[branch-name]` vers le dépôt distant `[remote-name]`.

N.B.

Toujours tirer les modifications des autres personnes et (éventuellement) les fusionner avec les vôtres avant de pouvoir pousser.

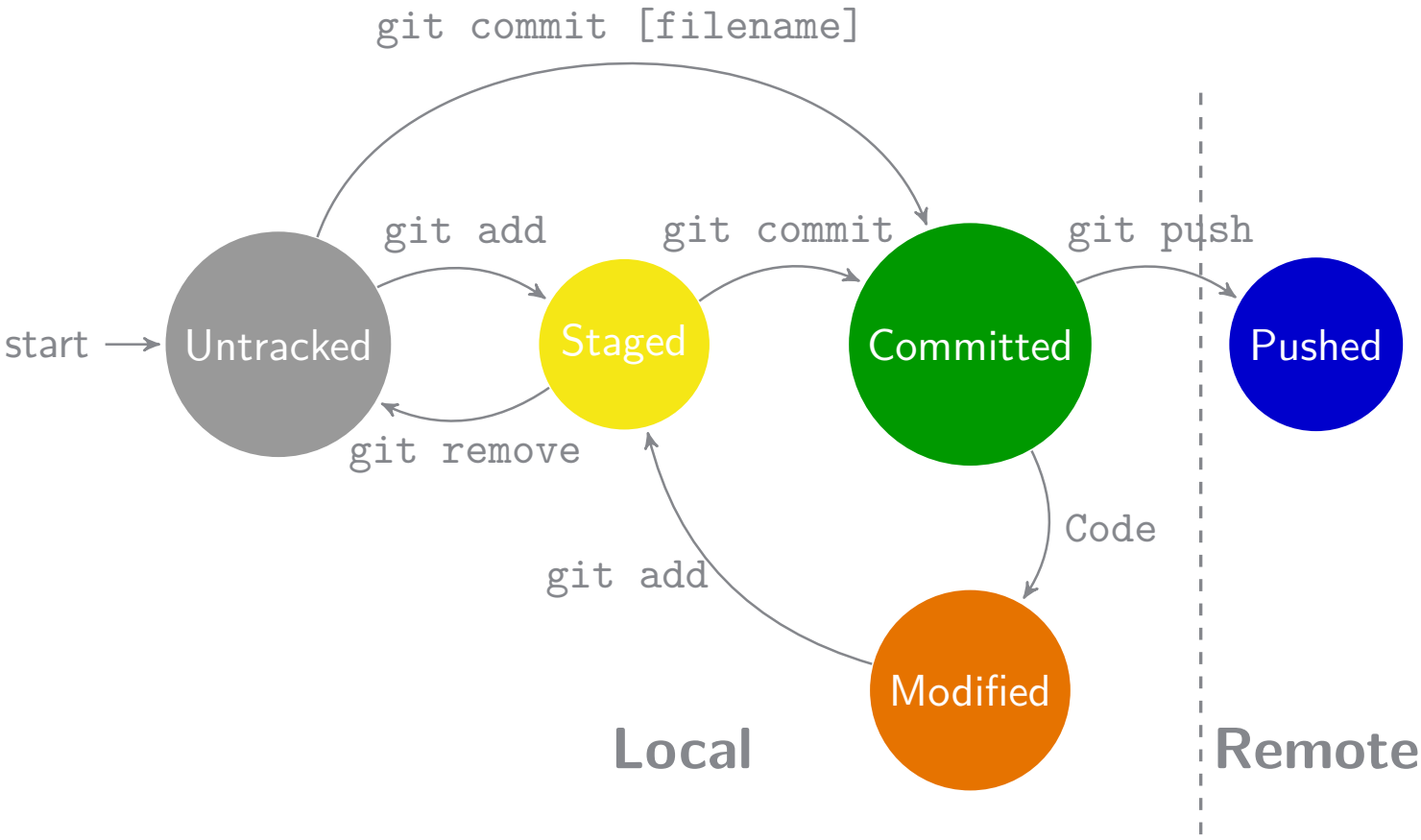
Adaptée de : Pro Git book by Scott Chacon and Ben Straub

Pour résumer



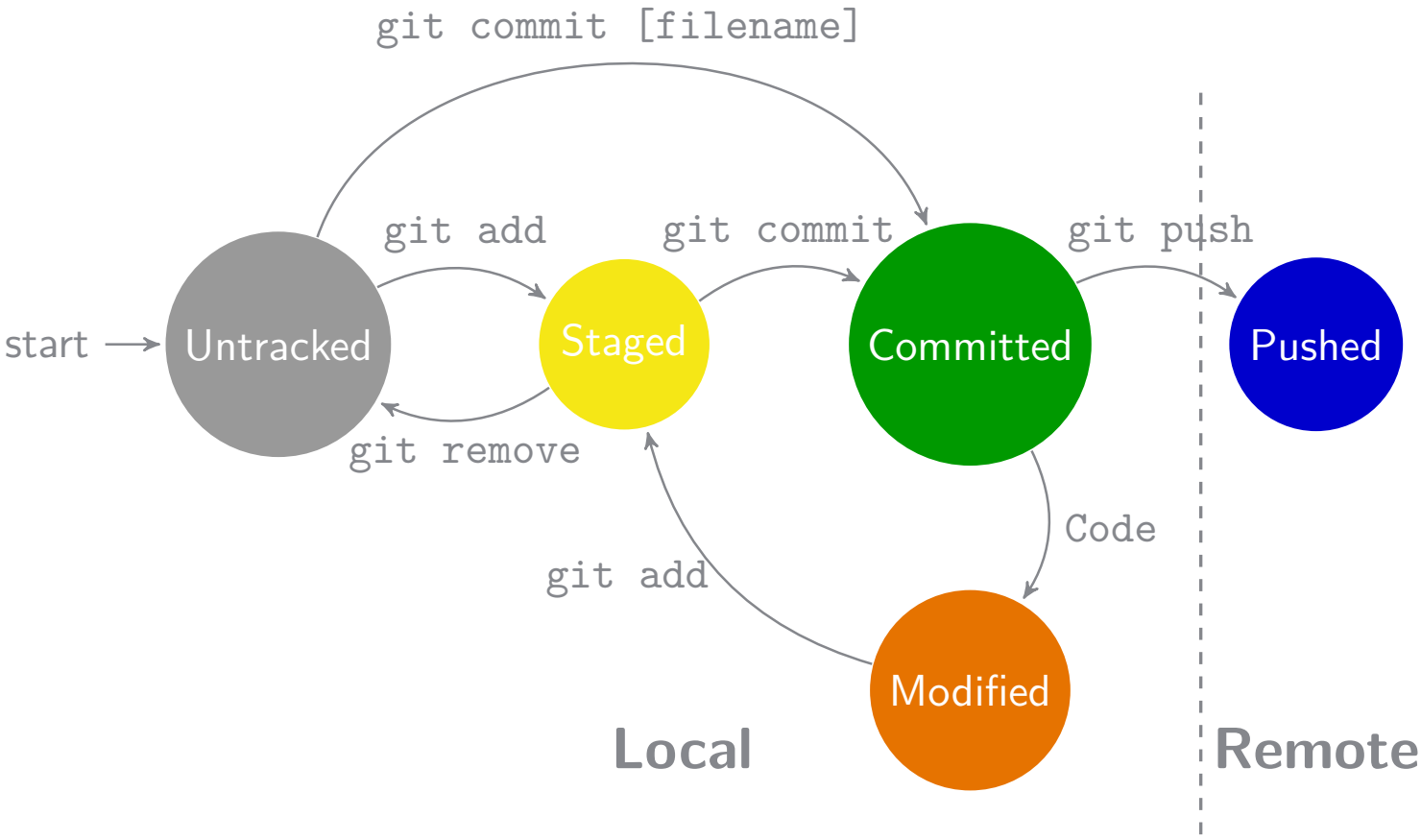
- Untracked = Pas encore sous contrôle de version
- Staged = Sous contrôle de version et prêt à la validation
- Modified = Fichier modifié après validation
- Committed = Modification validée
- Pushed = Modification transféré sur un dépôt distant (remote)

Pour résumer



- Untracked = Pas encore sous contrôle de version
- Staged = Sous contrôle de version et prêt à la validation
- Modified = Fichier modifié après validation
- Committed = Modification validée
- Pushed = Modification transféré sur un dépôt distant (remote)

Pour résumer



- `$ git add [file]`
  Ajoute `[file]` au système du suivi de version.
- `$ git rm (--cached) [file]`
  Retire `[file]` du système du suivi de version et garde (`--cached`)/supprime le fichier correspondant dans la copie locale.

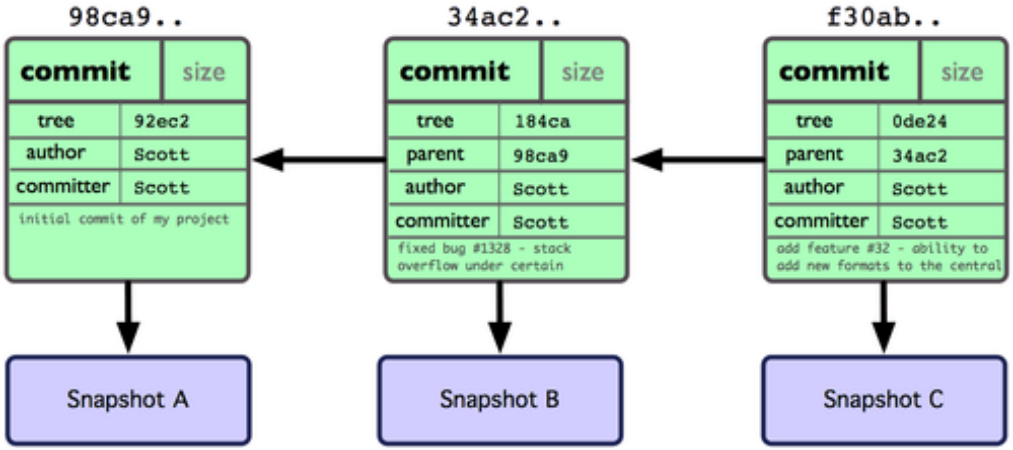
Source : Pro Git book by Scott Chacon and Ben Straub

ULB

UNIVERSITÉ LIBRE DE BRUXELLES

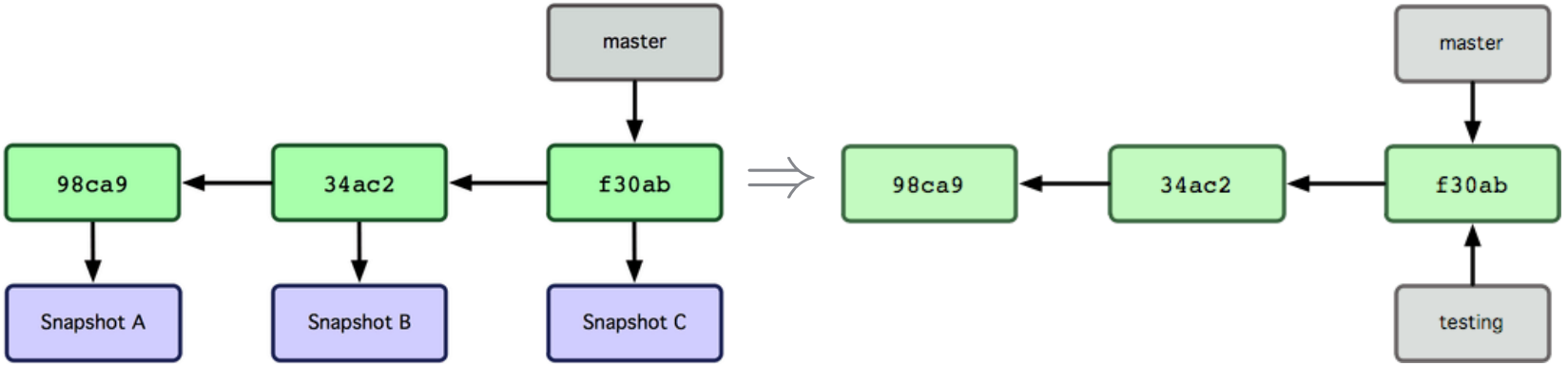
Git branching

Details d'un commit



Source : Pro Git book by Scott Chacon and Ben Straub

Git Branching (1)

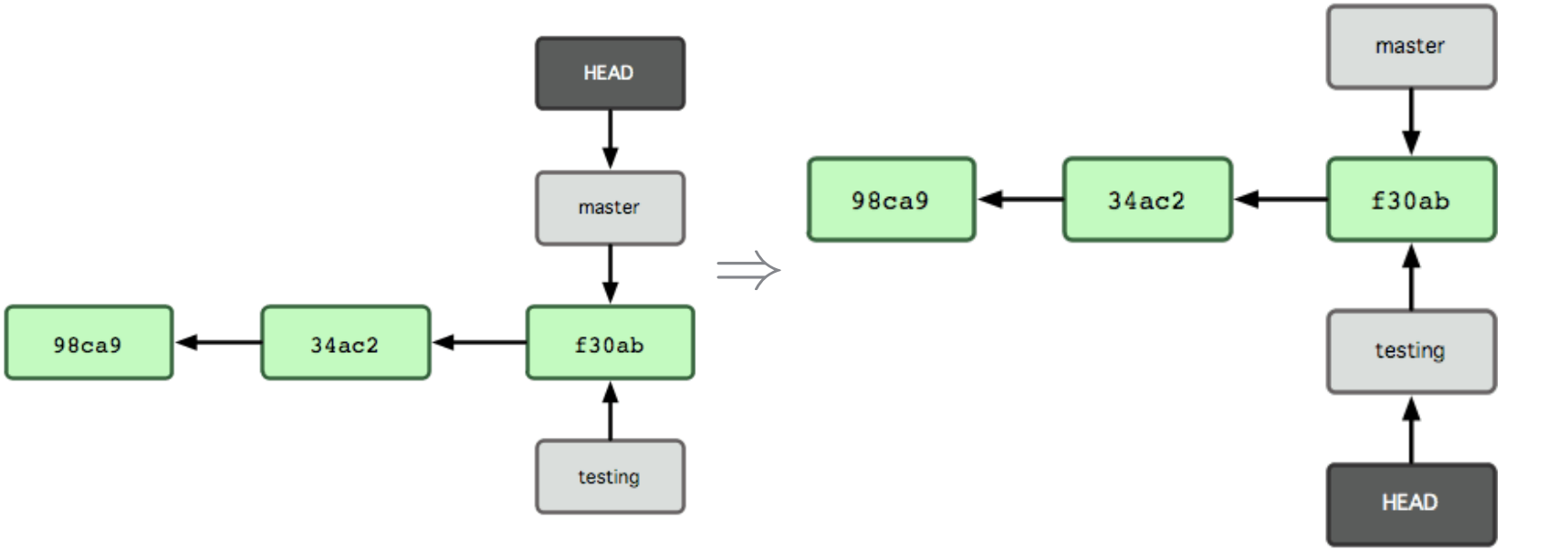



```
$ git branch testing
```

- Branche Git ≡ Pointeur léger et déplaçable vers un des commits.
- `master` ≡ Branche par défaut

Source : Pro Git book by Scott Chacon and Ben Straub

Git Branching (2)

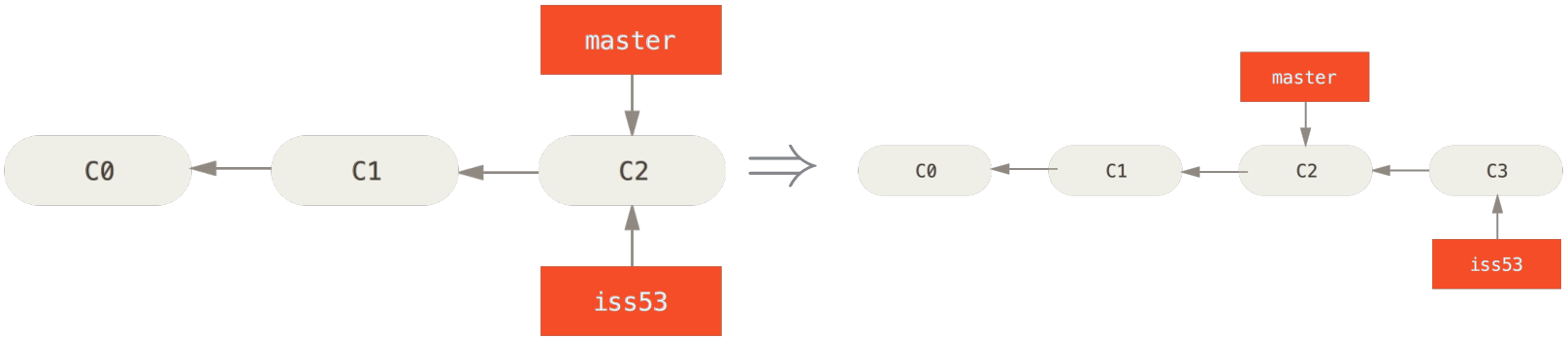



```
$ git checkout testing
```

- On travaille sur une seule branche à la fois.
- `HEAD` ≡ Pointeur special Git vers la branche courante.
- `HEAD` avance automatiquement a chaque commit.

Source : Pro Git book by Scott Chacon and Ben Straub

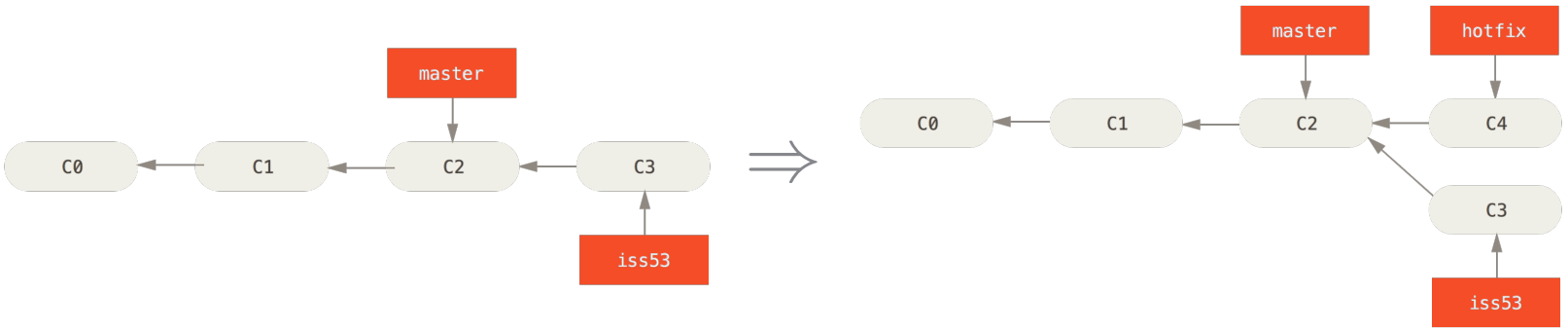
Git Merge (1)




```
$ git checkout -b iss53
# Modifications au projet dans la branche iss53
$ git commit -a -m "[probleme 53]"
```

Source : Pro Git book by Scott Chacon and Ben Straub

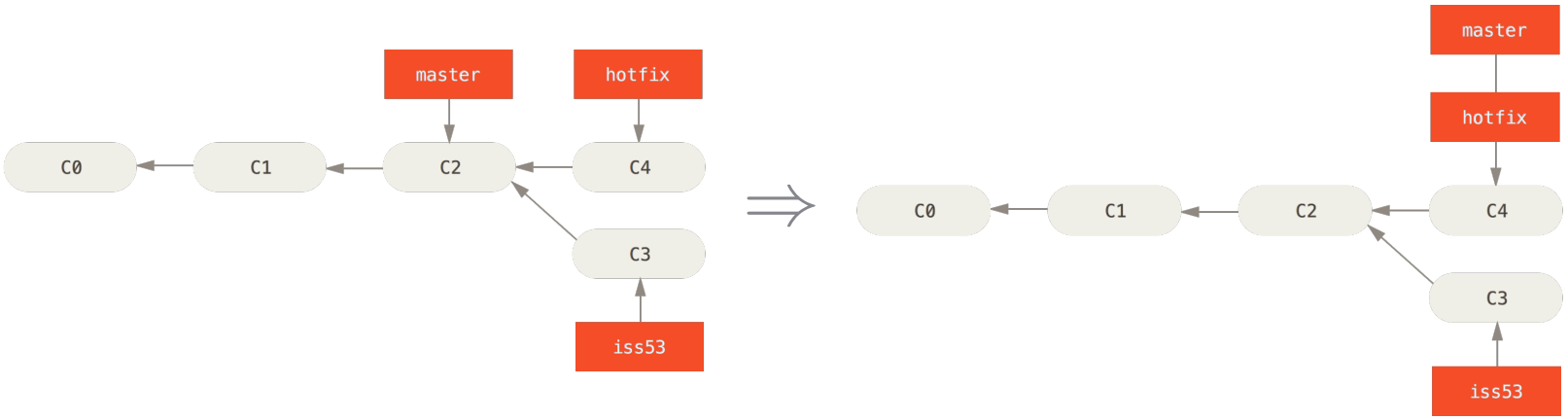
Git Merge (2)




```
$ git checkout -b hotfix
# Modifications au projet dans la branche hotfix
$ git commit -a -m "correction de l'adresse email
    incorrecte"
```

Source : Pro Git book by Scott Chacon and Ben Straub

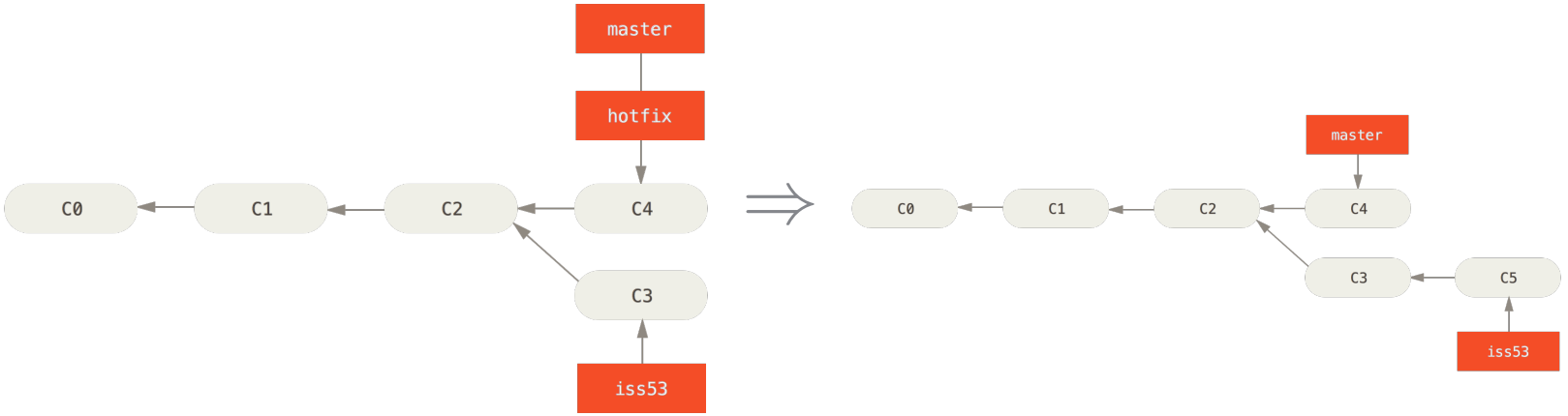
Git Merge (3)




```
$ git checkout master
$ git merge hotfix
Updating f42c576..3a0874c
Fast-forward
 index.html | 2 ++
 1 file changed, 2 insertions(+)
```

Source : Pro Git book by Scott Chacon and Ben Straub

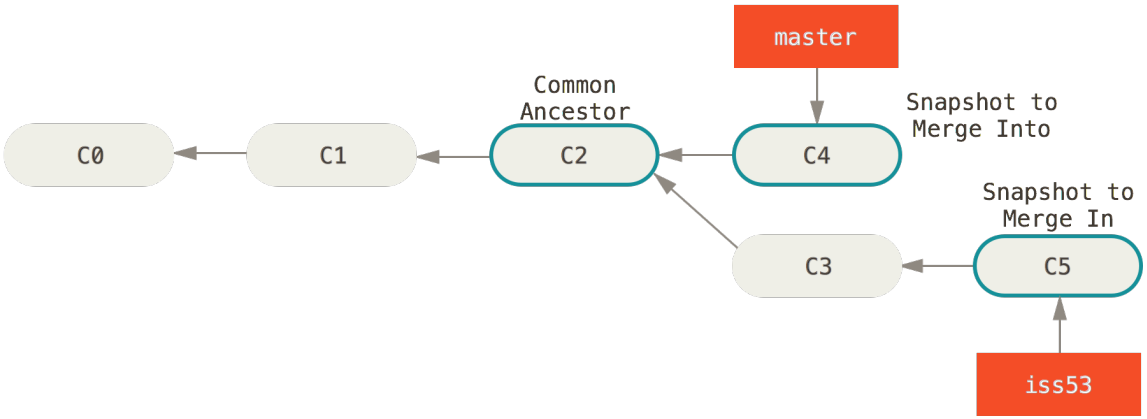
Git Merge (4)




```
$ git branch -d hotfix #Supprime branche hotfix
$ git checkout iss53
# Modifications au projet dans la branche iss53
$ git commit -a -m 'Nouveau pied de page termine [issue
    53]'
[iss53 ad82d7a] Nouveau pied de page termine [issue 53]
1 file changed, 1 insertion(+)
```

Source : Pro Git book by Scott Chacon and Ben Straub

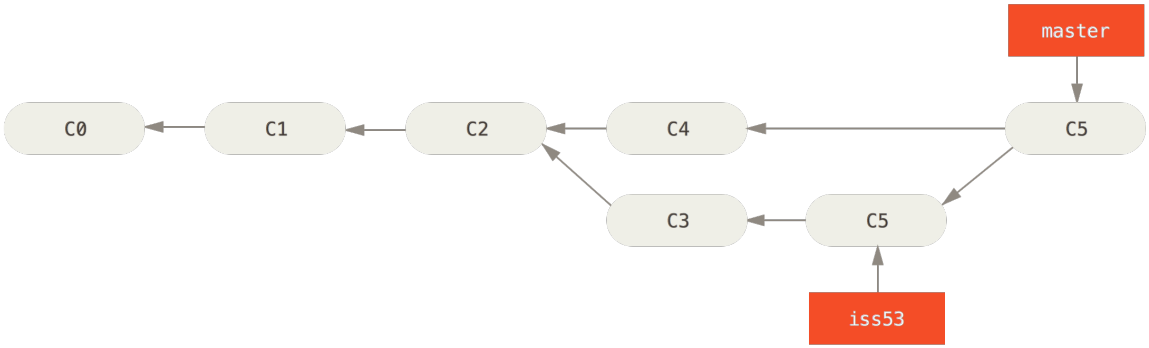
Git Merge (5)




```
$ git checkout master
Switched to branch 'master'
$ git merge iss53
Merge made by the 'recursive' strategy.
README |    1 +
1 file changed, 1 insertion(+)
```

Source : Pro Git book by Scott Chacon and Ben Straub

Git Merge (5)




```
$ git checkout master
Switched to branch 'master'
$ git merge iss53
Merge made by the 'recursive' strategy.
README |    1 +
1 file changed, 1 insertion(+)
```

Source : Pro Git book by Scott Chacon and Ben Straub

Géstion des conflits

```
#1. Demande funsion
$ git merge prob53
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the
     result.
#2. Verification conflict
$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
#3. Resolution du conflit grace aux outils mergetools
$ git mergetool
#4. Verification resolution conflict
$ git status
On branch master
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)
#5. Commit de la resolution
$ git commit
```

ULB

UNIVERSITÉ LIBRE DE BRUXELLES

Exercice 1

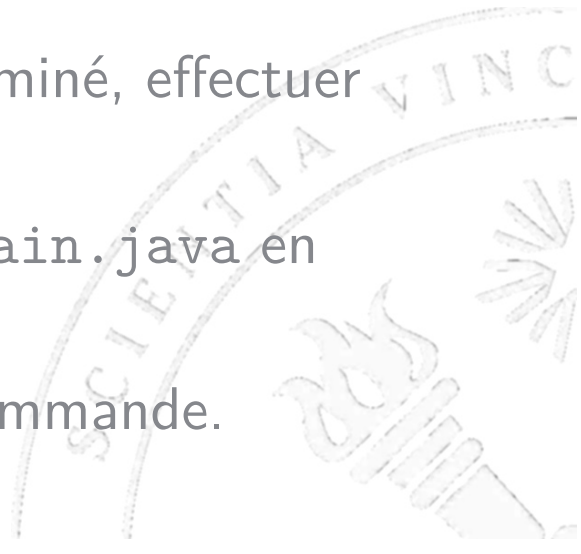
1. Cloner le dépôt Git distant `https://gitlab.com/jdestefani/ulb-infof307-tp1.git`
2. Modifier le fichier Java dans le dossier `Ex1` en ajoutant les informations requises.
3. Compiler le fichier Java en ligne de commande.
4. Executer le fichier en ligne de commande.
5. Effectuer un commit sur le dépôt Git local.

ULB

UNIVERSITÉ LIBRE DE BRUXELLES

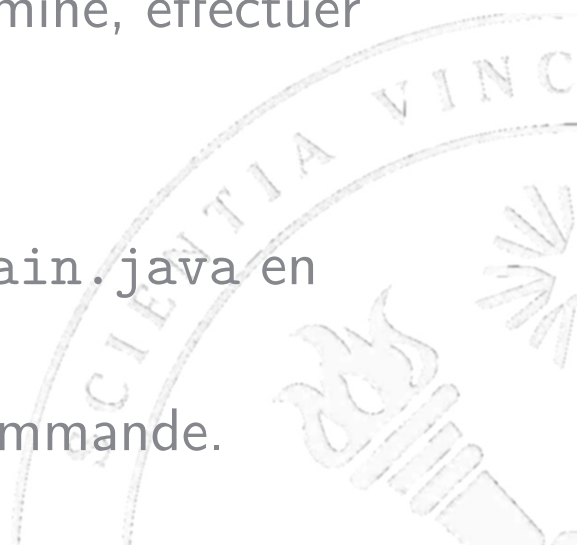
Exercice 2

1. Cloner le dépôt Git distant `https://gitlab.com/jdestefani/ulb-infof307-tp1.git`
2. Former un binôme avec votre voisin.
3. Pour chaque personne, sur le même ordinateur :
   - 3.1 Créer une branche dont le nom correspond à votre netID.
   - 3.2 Discuter avec le voisin les modifications à faire sur le code.
   - 3.3 Modifier le fichier `Calculette.java` dans le dossier `Ex2` selon les consignes.
   - 3.4 Effectuer un commit sur le dépôt Git local dans sa propre branche.
4. Une fois que les modifications auront été terminé, effectuer une fusion des deux branches.
5. Compiler les fichiers `Calculette.java` et `Main.java` en ligne de commande.
6. Exécuter le fichier `Main.java` en ligne de commande.



Exercice 3

1. Cloner le dépôt Git distant `https://gitlab.com/jdestefani/ulb-infof307-tp1.git`
2. Former un binôme avec votre voisin.
3. Pour chaque personne, sur le même ordinateur :
   - 3.1 Créer une branche dont le nom correspond à votre netID.
   - 3.2 Modifier le fichier `Calculette.java` dans le dossier `Ex2` dans le même endroit que le voisin, mais de façon différente.
   - 3.3 Effectuer un commit sur le dépôt Git local dans sa propre branche.
4. Une fois que les modifications auront été terminé, effectuer une fusion des deux branches.
5. Résoudre le conflict.
6. Compiler les fichiers `Calculette.java` et `Main.java` en ligne de commande.
7. Exécuter le fichier `Main.java` en ligne de commande.



ULB

UNIVERSITÉ LIBRE DE BRUXELLES

Exercice 4

Amusez-vous !

`http://learngitbranching.js.org/`