Les outils de gestion de versions

Hakim MOKEDDEM
École nationale Supérieure d'Informatique

Plan

- Généralités sur les outils de gestion de versions
- La gestion de versions avec le Git
  - Les bases du Git
  - Utilisation des branches
  - Fusion des branches
- La collaboration sur GitHub
- Bonnes pratiques d'utilisation du Git et GitHub

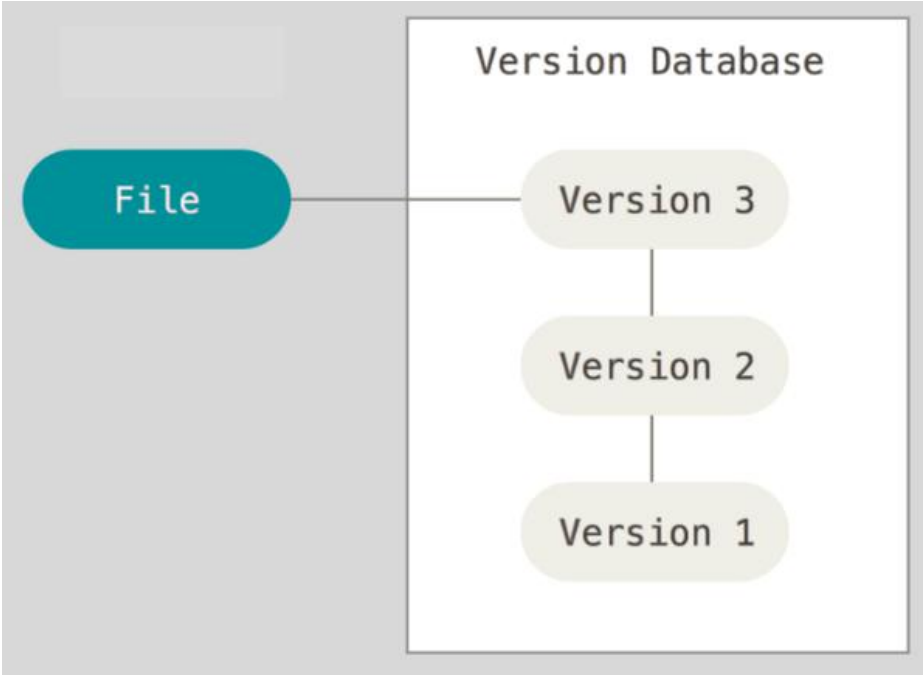
Plan

- Généralités sur les outils de gestion de versions
- La gestion de versions avec le Git
  - Les bases du Git
  - Utilisation des branches
  - Fusion des branches
- La collaboration sur GitHub
- Bonnes pratiques d'utilisation du Git et GitHub

Généralités sur les outils de gestion de versions

- **C'est quoi un outil de gestion de versions ?**

Un logiciel qui enregistre les modifications apportées à un fichier ou à un ensemble de fichiers au fil du temps.



Généralités sur les outils de gestion de versions

Pourquoi les outils de gestion de versions ?

- Stocker les différentes versions.
- Restaurer les différentes versions.
- Faciliter la compréhension des différentes versions.
- Faciliter le travail en équipe.

Généralités sur les outils de gestion de versions

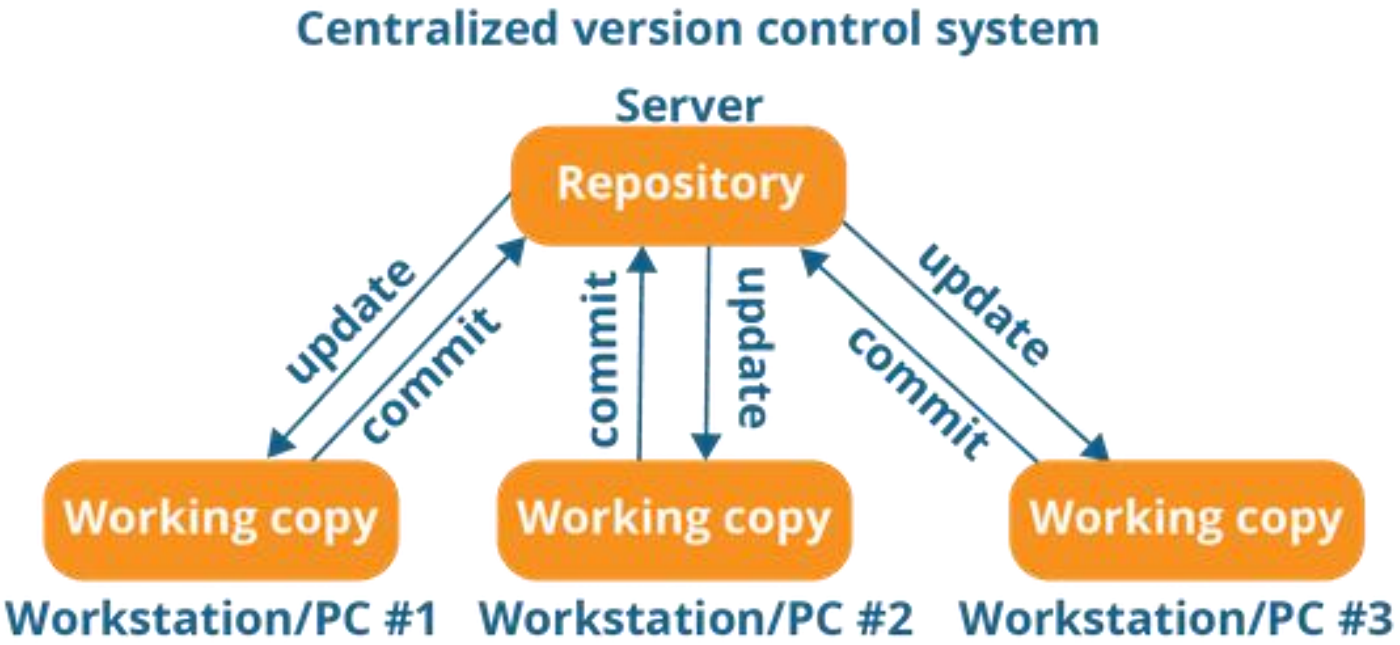
Quelles sont les architecture des outils de gestion de version ?

Gestion de Versions Centralisée

Gestion de Versions Distribuée

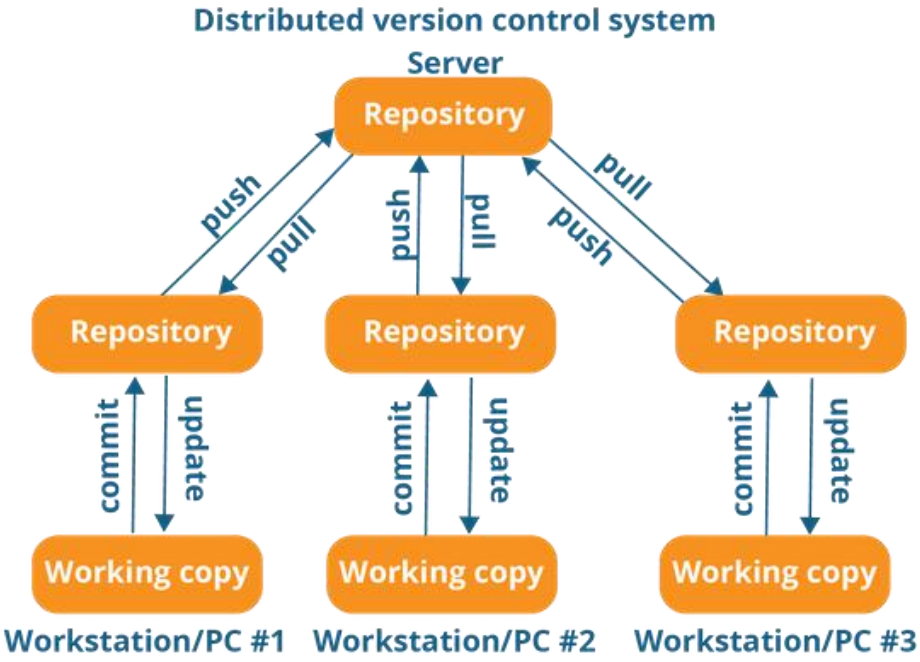
Gestion de versions centralisée Vs. Gestion de versions distribuée

- Une seule copie centrale du projet sur un serveur.
- Les développeurs valident les modifications sur le serveur.



Gestion de versions centralisée Vs. Gestion de versions distribuée

- Chaque développeur possède sa copie du projet.
- Les développeurs valident les modifications en local.
- Ils partagent les modifications sur le serveur.



Gestion de versions centralisée Vs. Gestion de versions distribuée

Outils de gestion de versions centralisée

Outils de gestion de versions distribuée

Gestion de versions centralisée Vs. Gestion de versions distribuée

Avantages

Gestion de versions centralisée

- La version finale est centralisée.

Gestion de versions distribuée

- Mode offline pour valider les versions.
- Expérimentation facile de nouvelles fonctionnalités.

Gestion de versions centralisée Vs. Gestion de versions distribuée

Inconvénients

Gestion de versions centralisée

- Besoin de la connexion pour valider les versions.

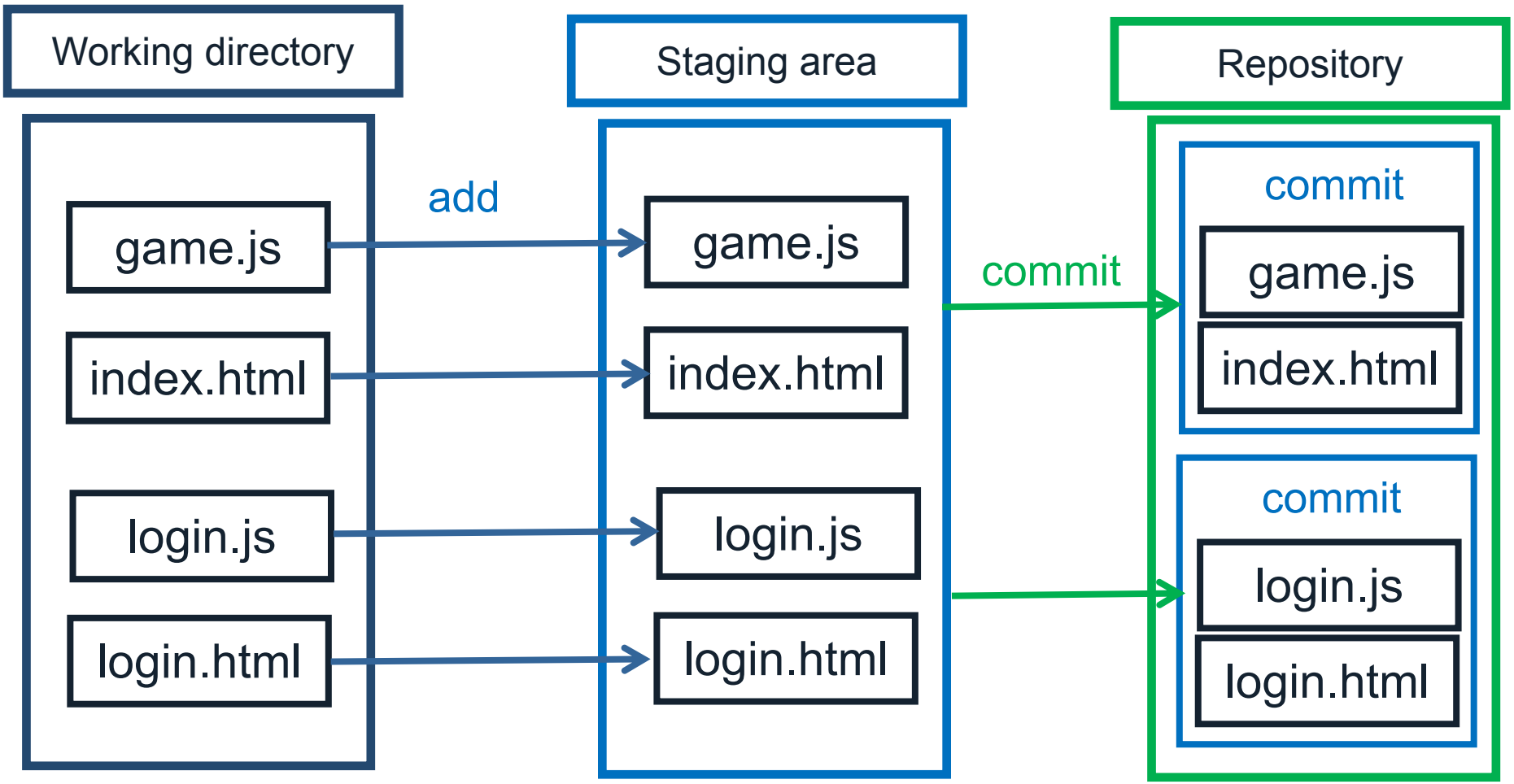
Gestion de versions distribuée

- Aucune vision claire sur la version finale.

Plan

- Généralités sur les outils de gestion de versions
- La gestion de versions avec le Git
  - Les bases du Git
  - Utilisation des branches
  - Fusion des branches
- La collaboration sur GitHub
- Bonnes pratiques d'utilisation du Git et GitHub

Les bases du Git



Les bases du Git

- **Working directory.** Le répertoire de travail en local.
- **Staging area .** Zone intermédiaire pour contrôler les versions à sauvegarder.
- **Repository.** Le répertoire Git (en local ou sur un serveur) où la base de fichiers est stockée.
- **Head.** La référence de la dernière version.

Les bases du Git
Les commandes de base du Git

Initialisation

Sauvegarde

Inspection

Restauration

Les bases du Git
Les commandes de base du Git

Initialisation

Sauvegarde

Inspection

Restauration

Les bases du Git
Les commandes de base du Git

- ***git init.*** Initialiser le répertoire du travail.
- ***git clone.*** Importer un *repository* en local.

Exemple

git clone https://github.com/libgit2/libgit2.git

- ***Le fichier .gitignore***. Utilisé pour ignoer des dossiers ou des fichiers.

Les bases du Git
Les commandes de base du Git

Initialisation

Sauvegarde

Inspection

Restauration

Les bases du Git
Les commandes de base du Git

- ***git add***

Mise à jour de la zone *staging area*.

Exemples

- ***git add file.txt*** *:* ajouter *file.txt* au *staging area*.
- ***git add -A*** : ajouter tous les fichiers au *staging area*.
- ***git add -u*** : ajouter tous les fichiers sauf les nouveaux.
- ***git add .*** : mettre à jour la zone *staging area* sans prise en compte des suppressions.

Les bases du Git
Les commandes de base du Git

- ***git commit***

Créer un nouveau commit.

Exemple

- ***git commit -m "Fix benchmarks for speed"***

Valider les changements avec un message.

Les bases du Git
Les commandes de base du Git

Initialisation

Sauvegarde

Inspection

Restauration

Les bases du Git
Les commandes de base du Git

- ***git status***

Afficher les fichiers modifiés du répertoire de travail et ceux *du staging area* à utiliser dans le prochain commit.

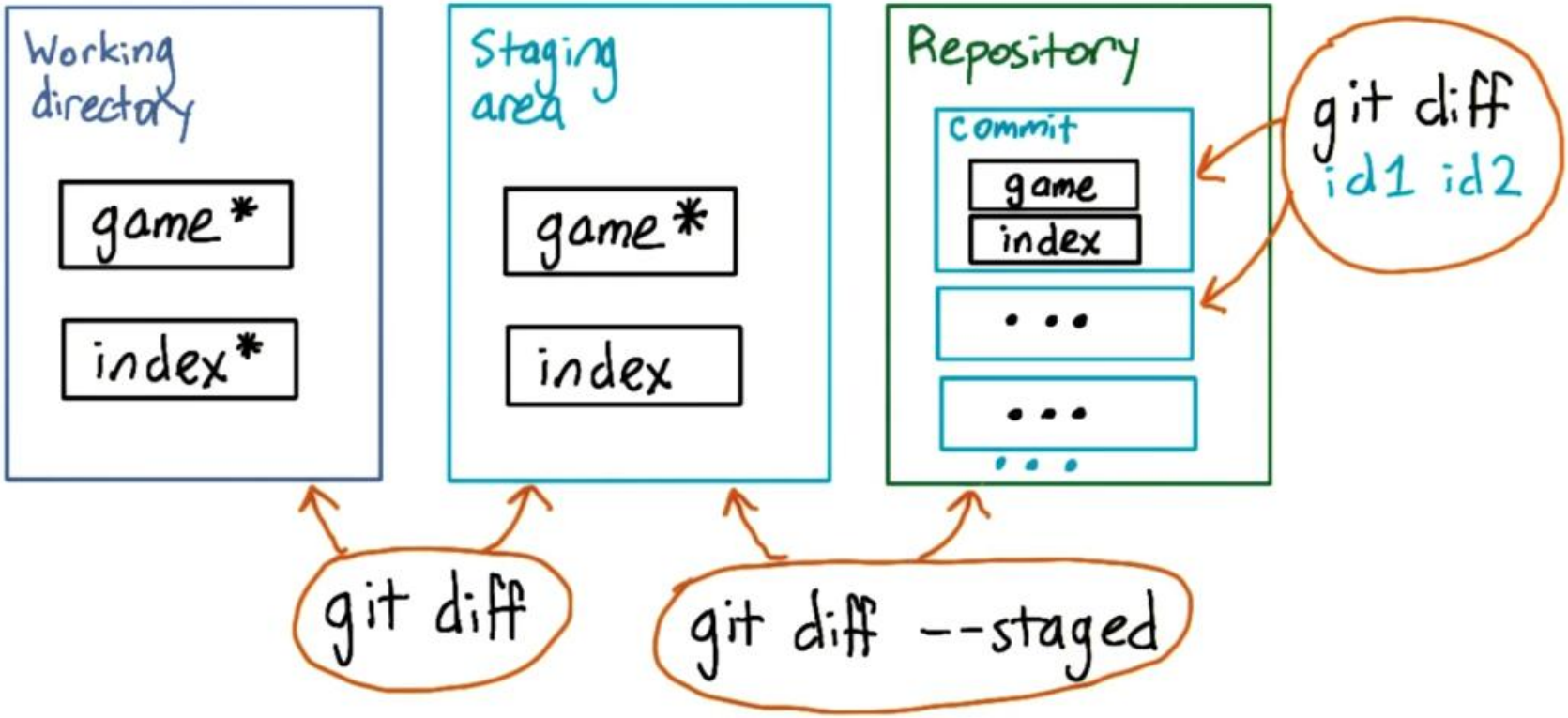
- ***git log***

Afficher l'historique des commits.

Les bases du Git
Les commandes de base du Git

- ***git diff***

Comparer les modifications



Les bases du Git
Les commandes de base du Git

Initialisation

Sauvegarde

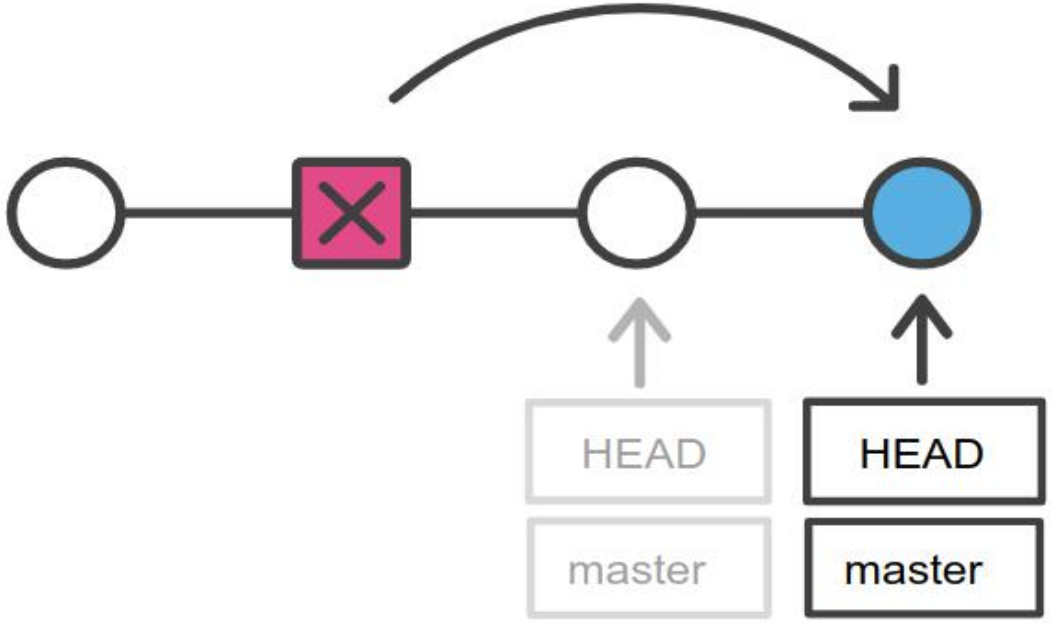
Inspection

Restauration

Les bases du Git
Les commandes de base du Git

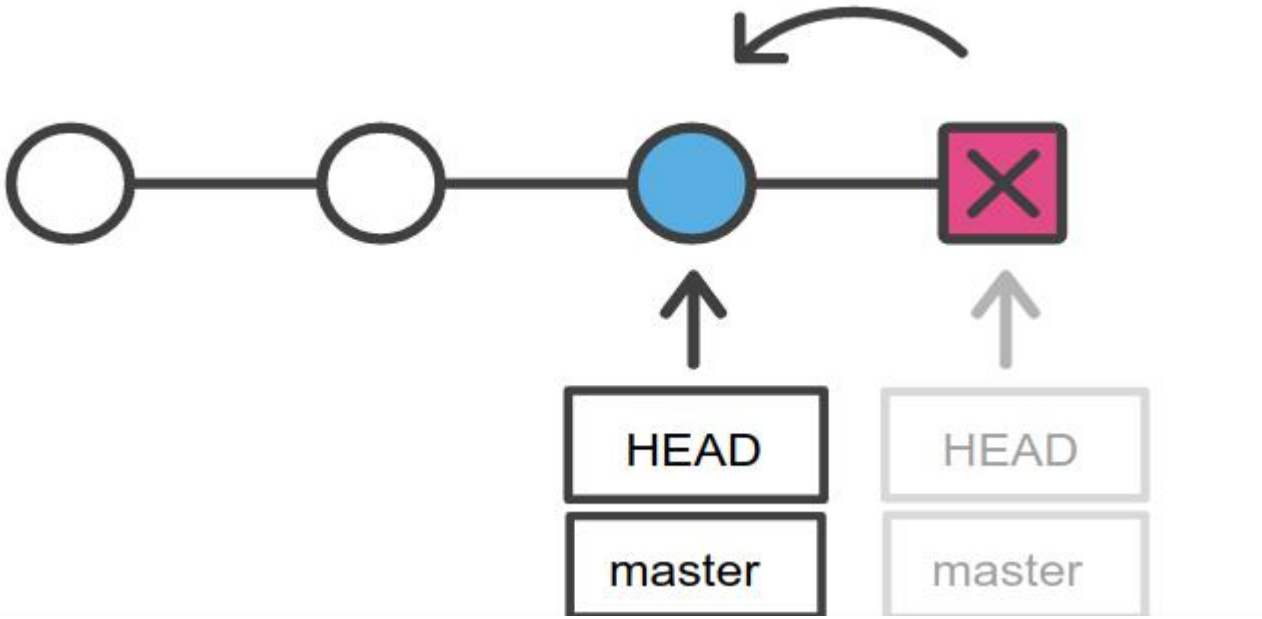
- ***git revert HEAD***

Une commande utile pout annuler les changements du dernier commit, en créant un nouveau commit.



Les bases du Git
Les commandes de base du Git

- ***git reset --hard id***
  - Pointer le *HEAD* sur le commit de l'identifiant *id*.
  - Tous les commits après le *HEAD* actuel seront perdus.



Plan

- Généralités sur les outils de gestion de versions
- La gestion de versions avec le Git
  - Les bases du Git
  - Utilisation des branches
  - Fusion des branches
- La collaboration sur GitHub
- Bonnes pratiques d'utilisation du Git et GitHub

Les branches dans le Git
Problèmes

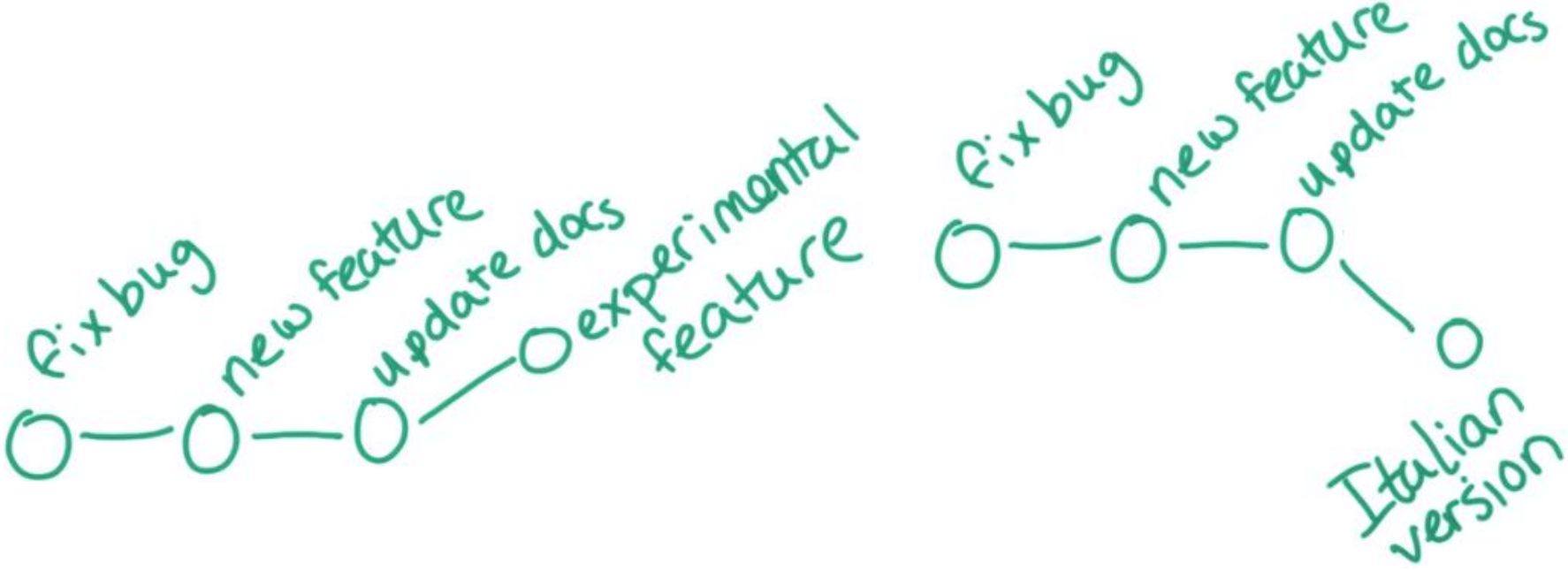
Trois versions sauvegardées du logiciel:

1. Traitement d'un bug.
2. Ajout d'une nouvelle fonctionnalité.
3. Mise à jour de la documentation.

Les branches dans le Git
Problèmes

Nouveaux besoins

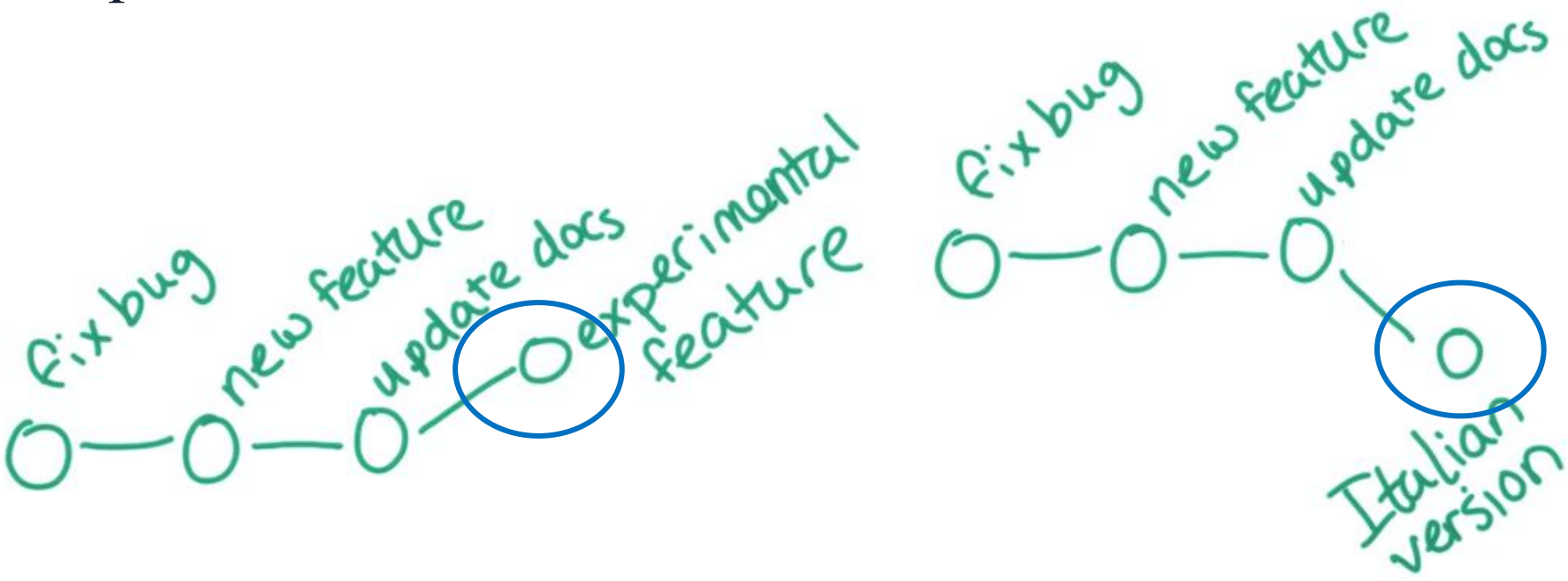
- Expérimenter une nouvelle fonctionnalité.
- Une version italienne du logiciel.



Les branches dans le Git
Problèmes

Problème 1

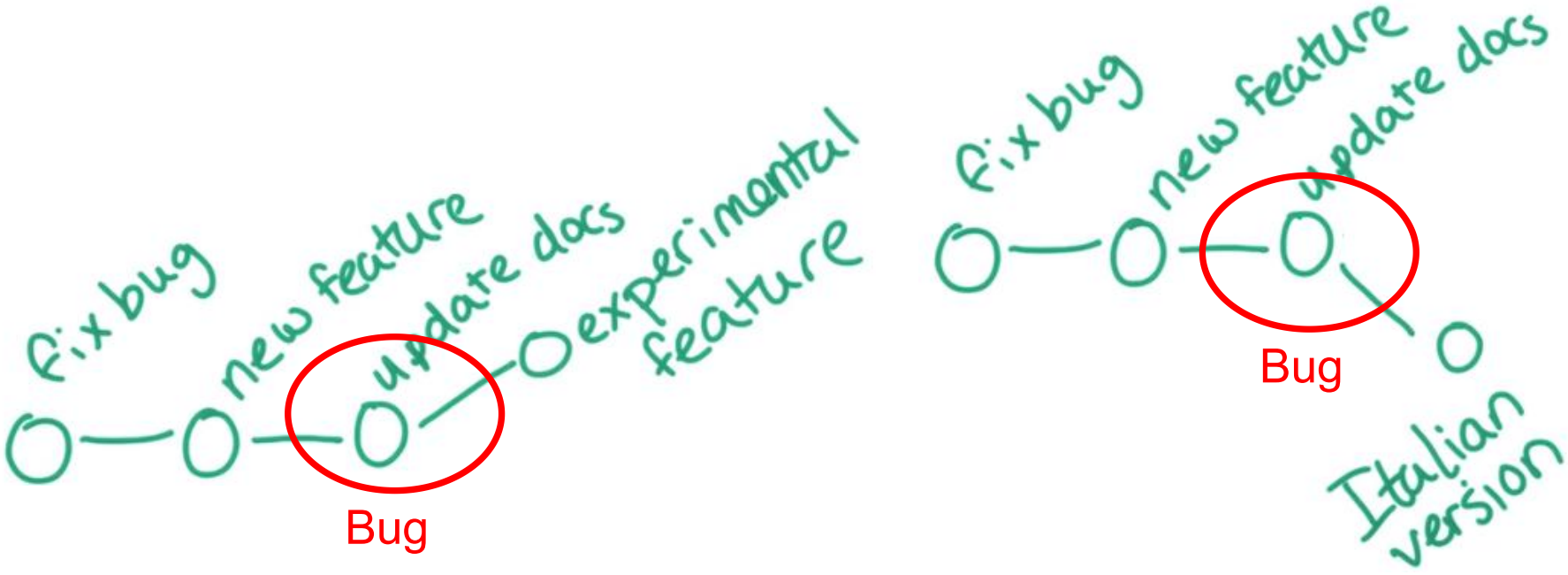
- Difficulté de développer les deux versions en parallèle.



Les branches dans le Git
Problèmes

Problème 2

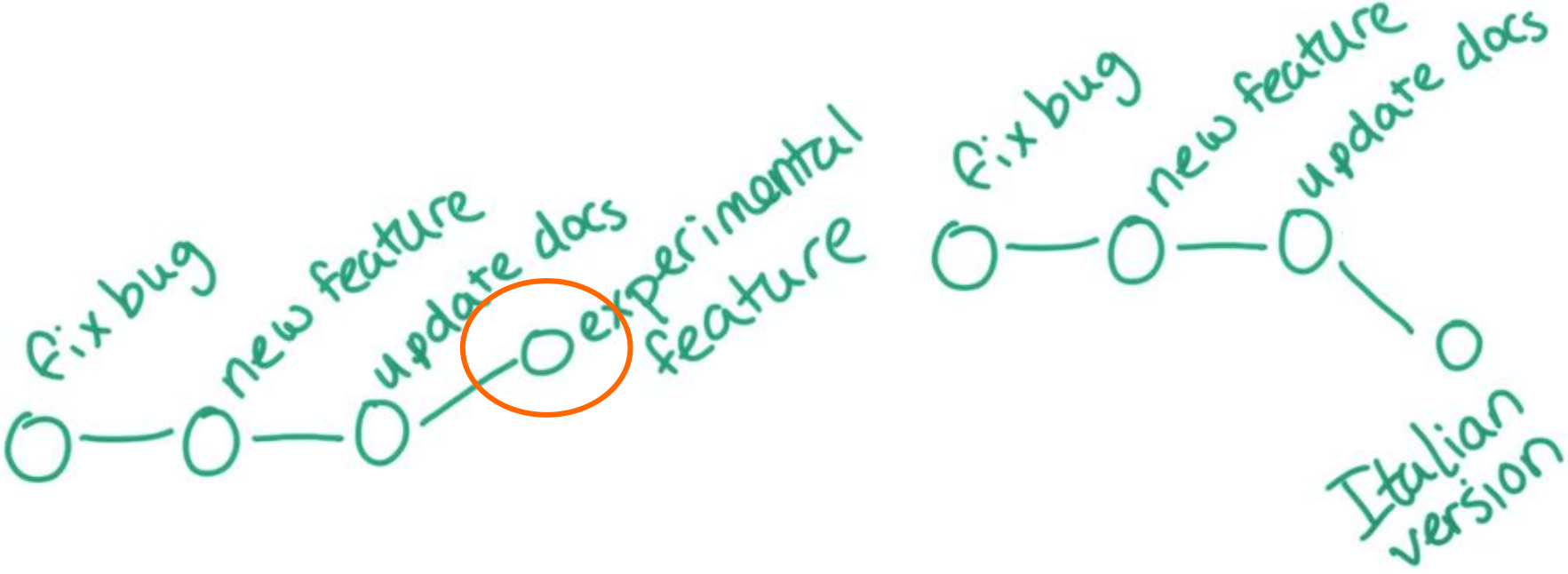
- Difficulté de traiter les nouveaux bugs détectés dans la version en production.



Les branches dans le Git
Problèmes

Problème 3

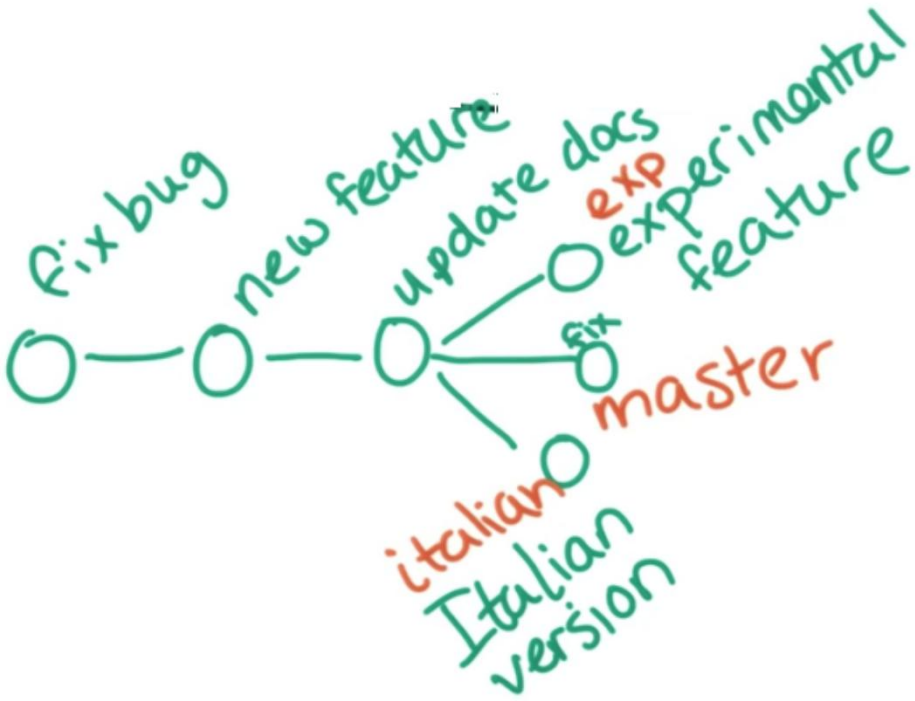
- Manque de flexibilité pour traiter l'instabilité de la fonctionnalité expérimentale.



Les branches dans le Git
Utilisation des branches

- **Solution.** Utilisation des branches
- **C'est quoi une branche ?**

Une ligne indépendante de développement.



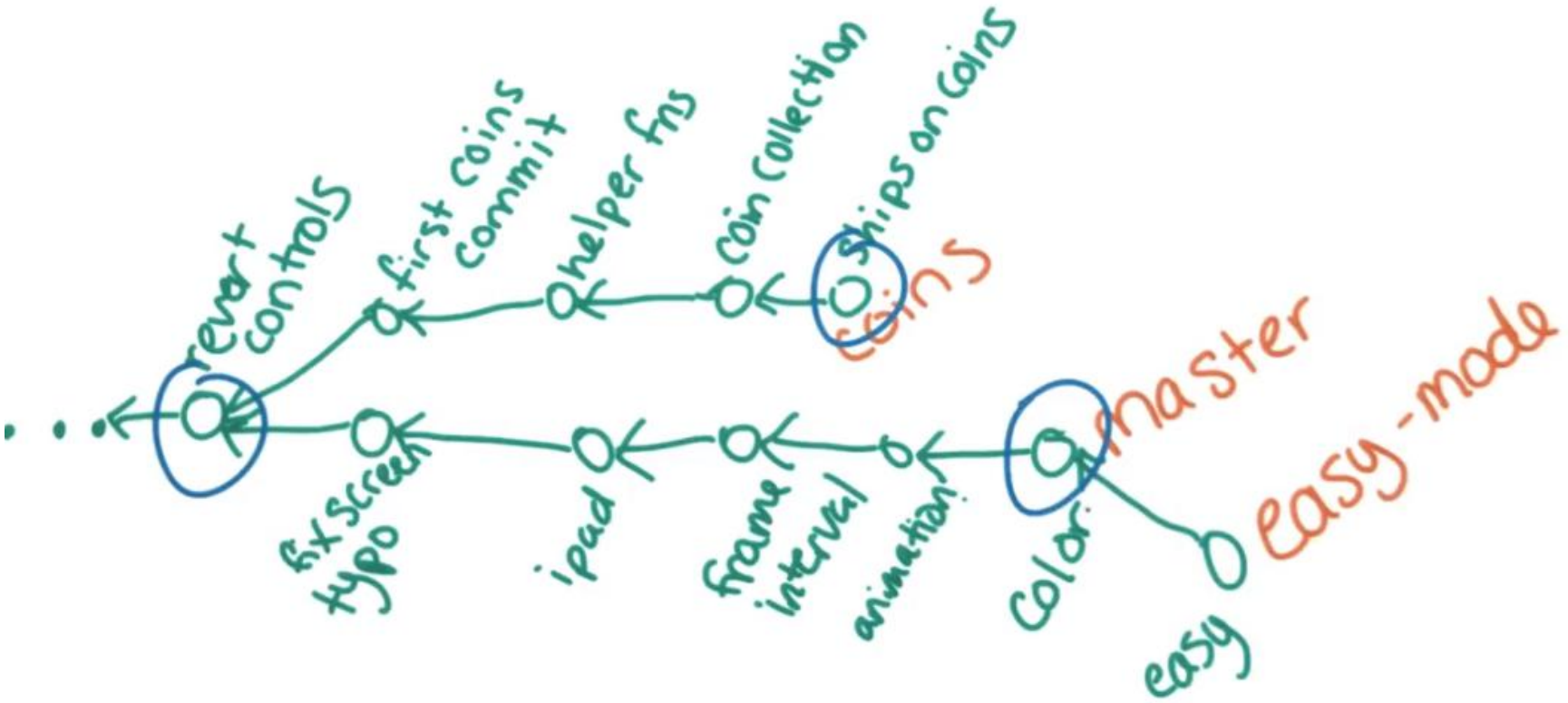
Les branches dans le Git
Les commandes de base

- ***git branch exp.*** Créer une nouvelle branche *exp*.
- ***git checkout exp.*** Pointer sur la branche *exp*.
- ***git checkout -b exp.*** Créer et pointer sur la nouvelle branche *exp*.
- ***git branch.*** Afficher la liste des branches.
- ***git branch -d exp.*** Supprimer la branche *exp*.

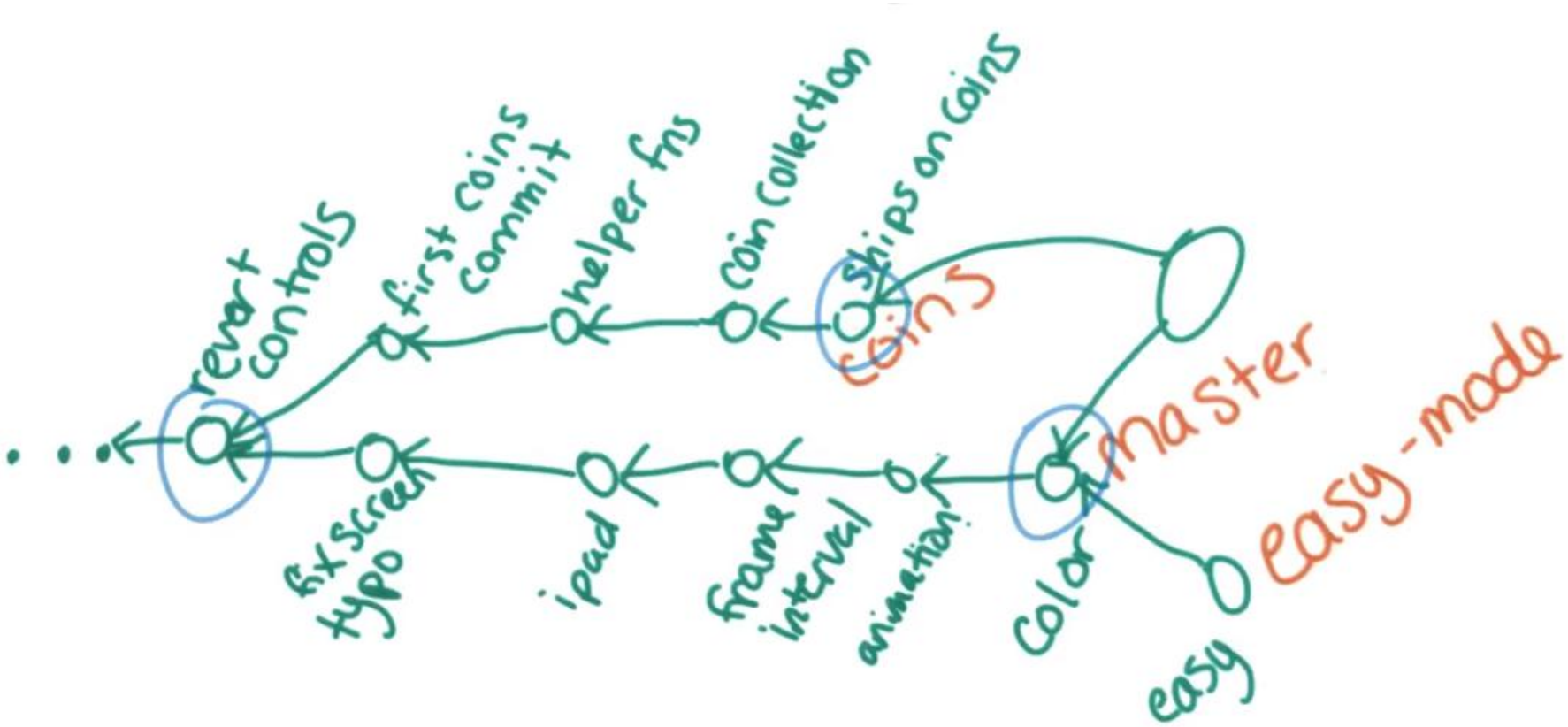
Plan

- Généralités sur les outils de gestion de versions
- La gestion de versions avec le Git
  - Les bases du Git
  - Utilisation des branches
  - Fusion des branches
- La collaboration sur GitHub
- Bonnes pratiques d'utilisation du Git et GitHub

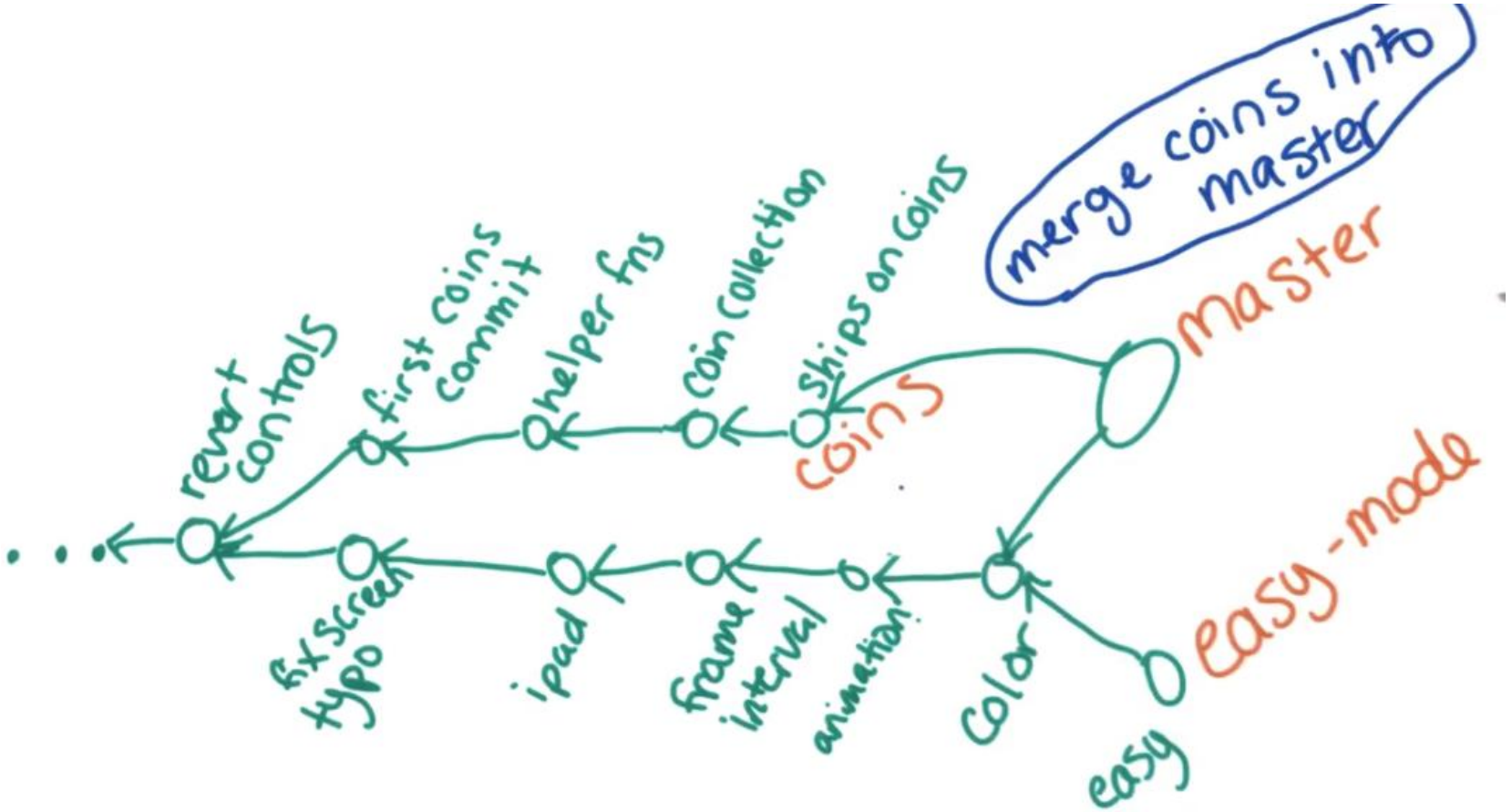
La fusion des branches
Exemple



La fusion des branches
Exemple



La fusion des branches
Exemple



La fusion des branches
Les commandes de base

En deux étapes:

- ***git checkout master***

Pointer dans la branche master.

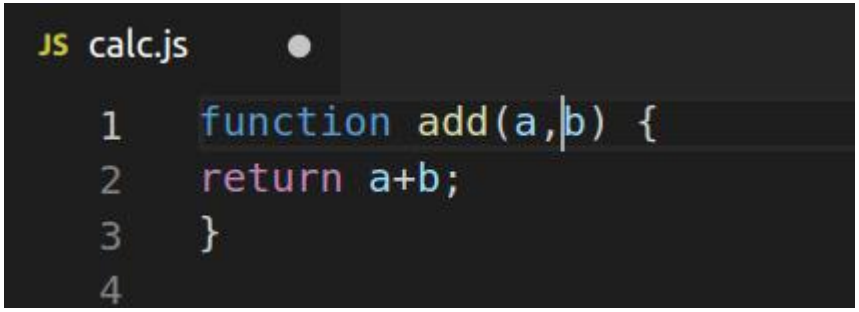
- ***git merge coins -m "message de fusion"***

Fusionner les branches *master* et *coins,* dans la branche master.

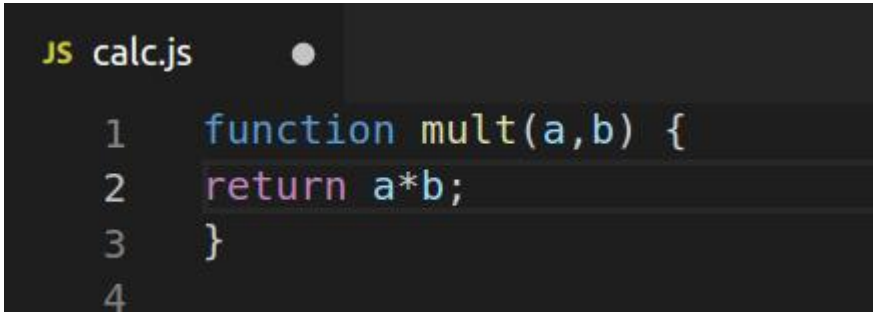
La fusion des branches
Gestion des conflits

C'est quoi un conflit ?

- Deux branches distinctes ont modifié la même ligne dans un fichier.
- Détecté après une tentative de fusion de branches.



Développeur 1



Développeur 2

La fusion des branches
Gestion des conflits

Comment traiter un conflit ?

- Modifier le fichier manuellement: garder les deux modifications ou garder une modification et supprimer l'autre.
- Ajouter le fichier au *staging area* et lancer un commit.

Plan

- Généralités sur les outils de gestion de versions
- La gestion de versions avec le Git
  - Les bases du Git
  - Utilisation des branches
  - Fusion des branches
- La collaboration sur GitHub
- Bonnes pratiques d'utilisation du Git et GitHub

La collaboration sur GitHub

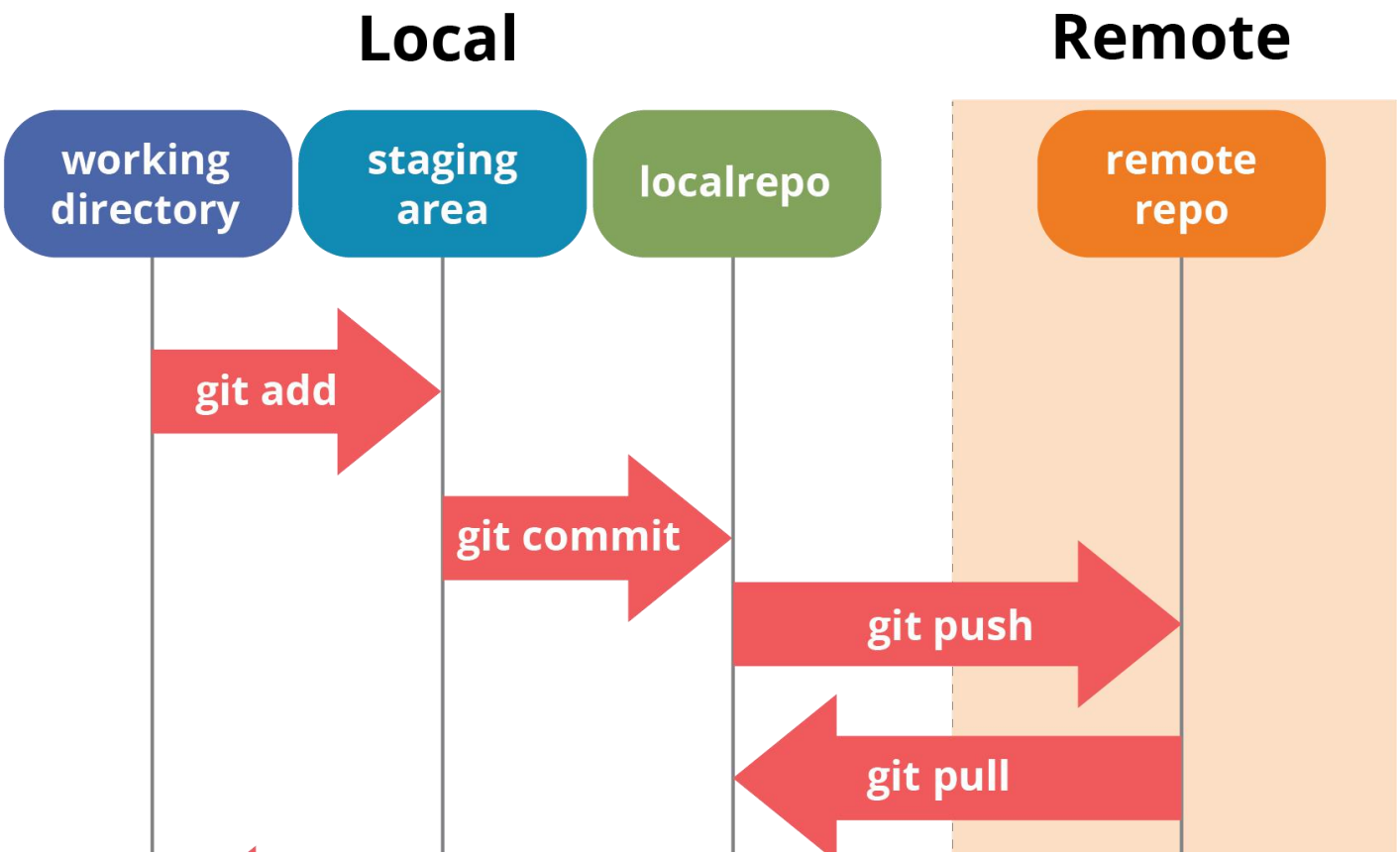
- **C'est quoi le GitHub ?**

Un service web d'hébergement et de gestion de développement de logiciels, utilisant le Git.

- **Quelles sont les alternatives du GitHub ?**

GitLab

La collaboration sur GitHub
Architecture d'utilisation



La collaboration sur GitHub
Les commandes de base

1. **Importer un repository GitHub**

git clone https://github.com/libgit2/libgit2.git

2. **Ajouter un remote en local**
    1. Créer un *repository* sur *GitHub*
    2. Lancer la commande

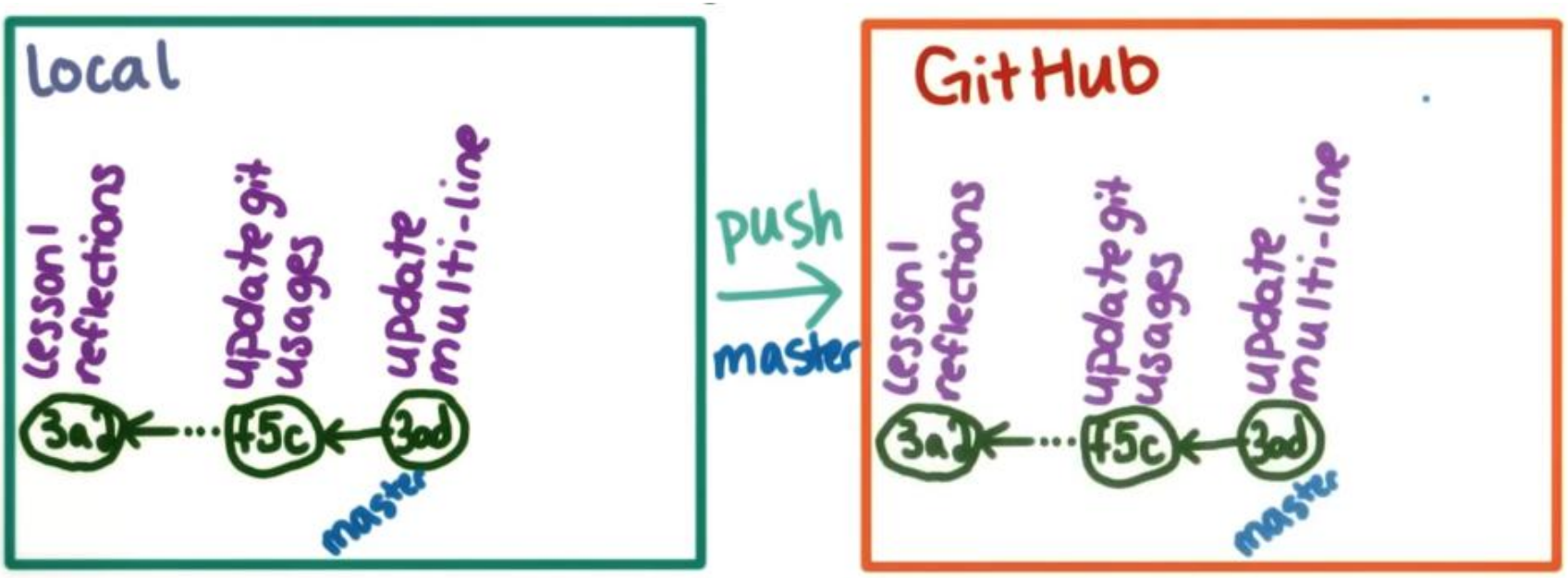
git remote add remote_name url_repository

Exemple

git remote add origin https://github.com/libgit2/libgit2.git

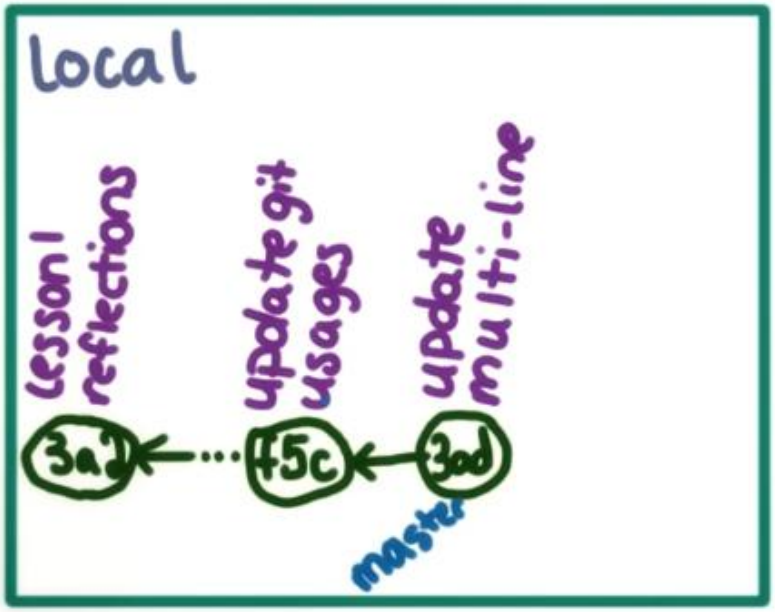
La collaboration sur GitHub
Les commandes de base

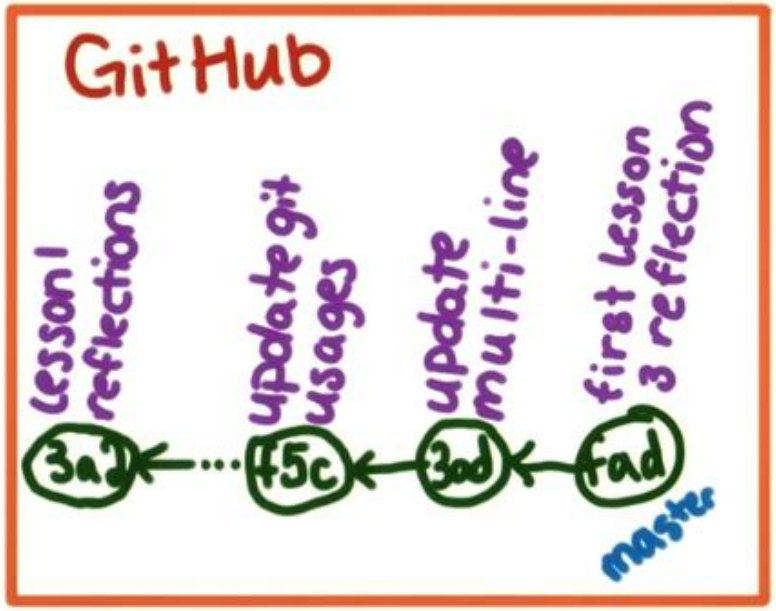
- ***git push origin master.*** Envoyer les commits de la branche master du local vers le serveur.



La collaboration sur GitHub
Les commandes de base

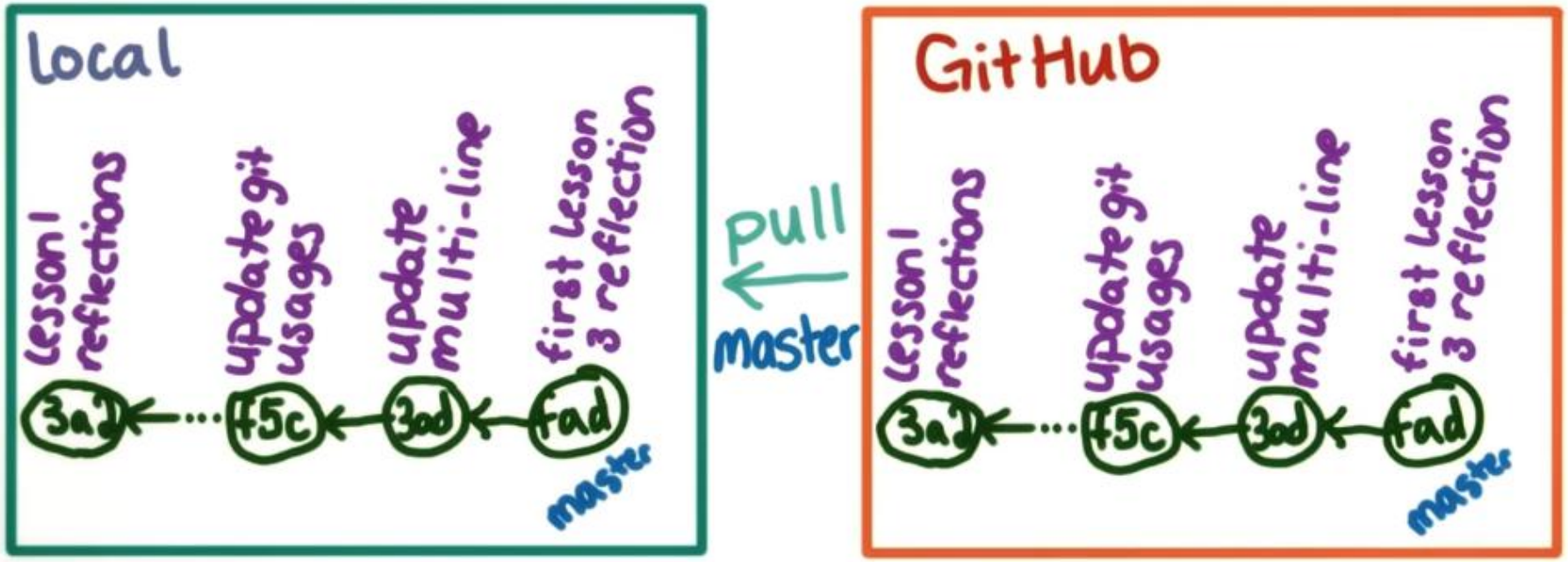
- ***git pull origin master.*** Envoyer les commits de la branche master du serveur vers le *repository* local.





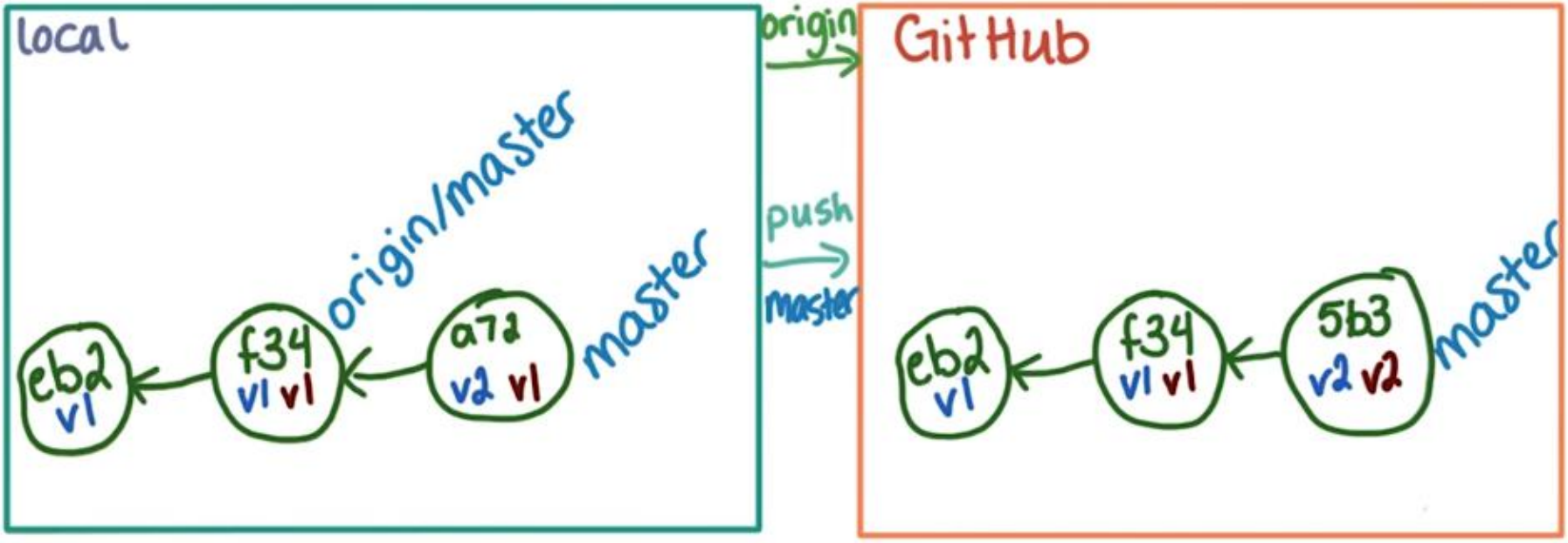
La collaboration sur GitHub
Les commandes de base

- ***git pull origin master.*** Envoyer les commits de la branche master du serveur vers le *repository* local.



La collaboration sur GitHub
Les conflits de collaboration

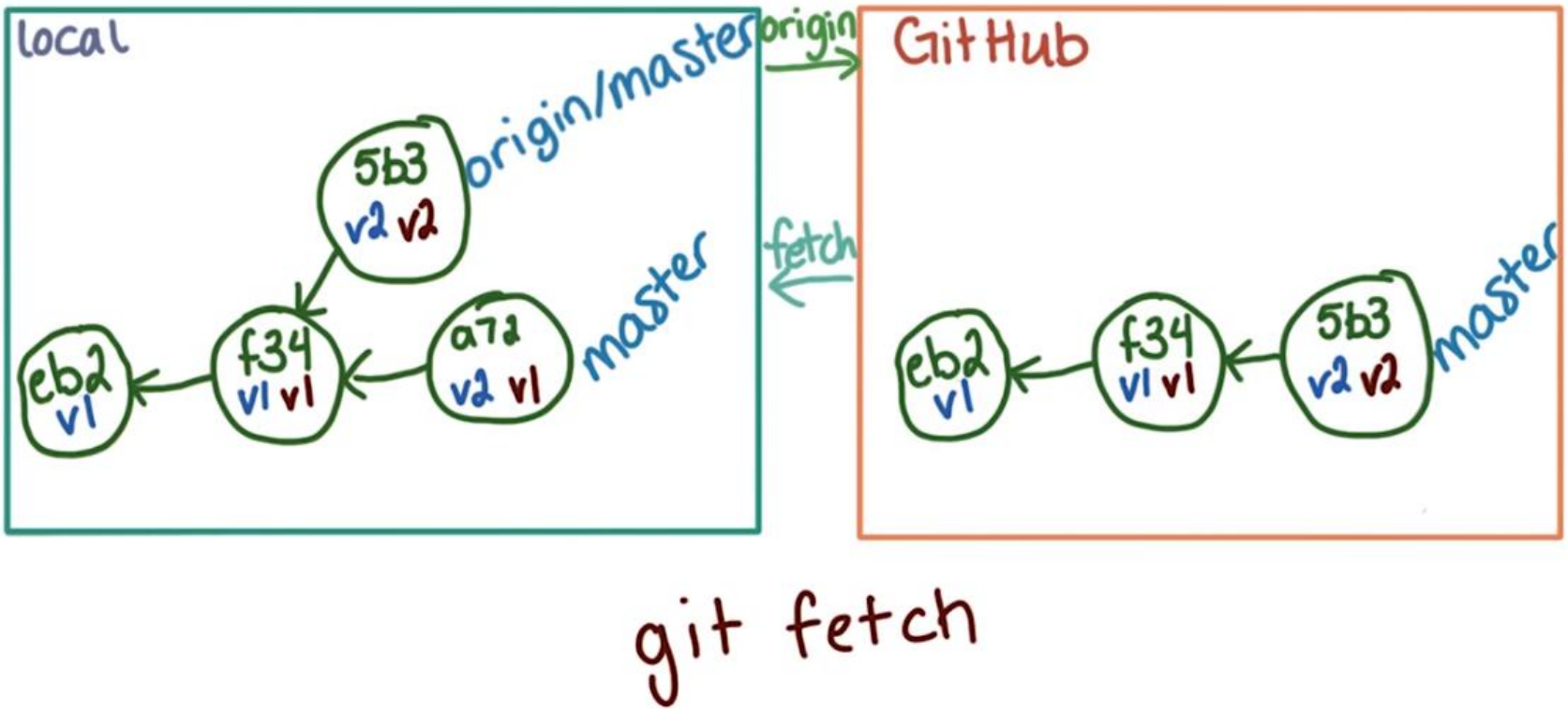
Problème. Deux versions différentes sur la même branche.



La collaboration sur GitHub
Les conflits de collaboration

- ***git fetch origin***

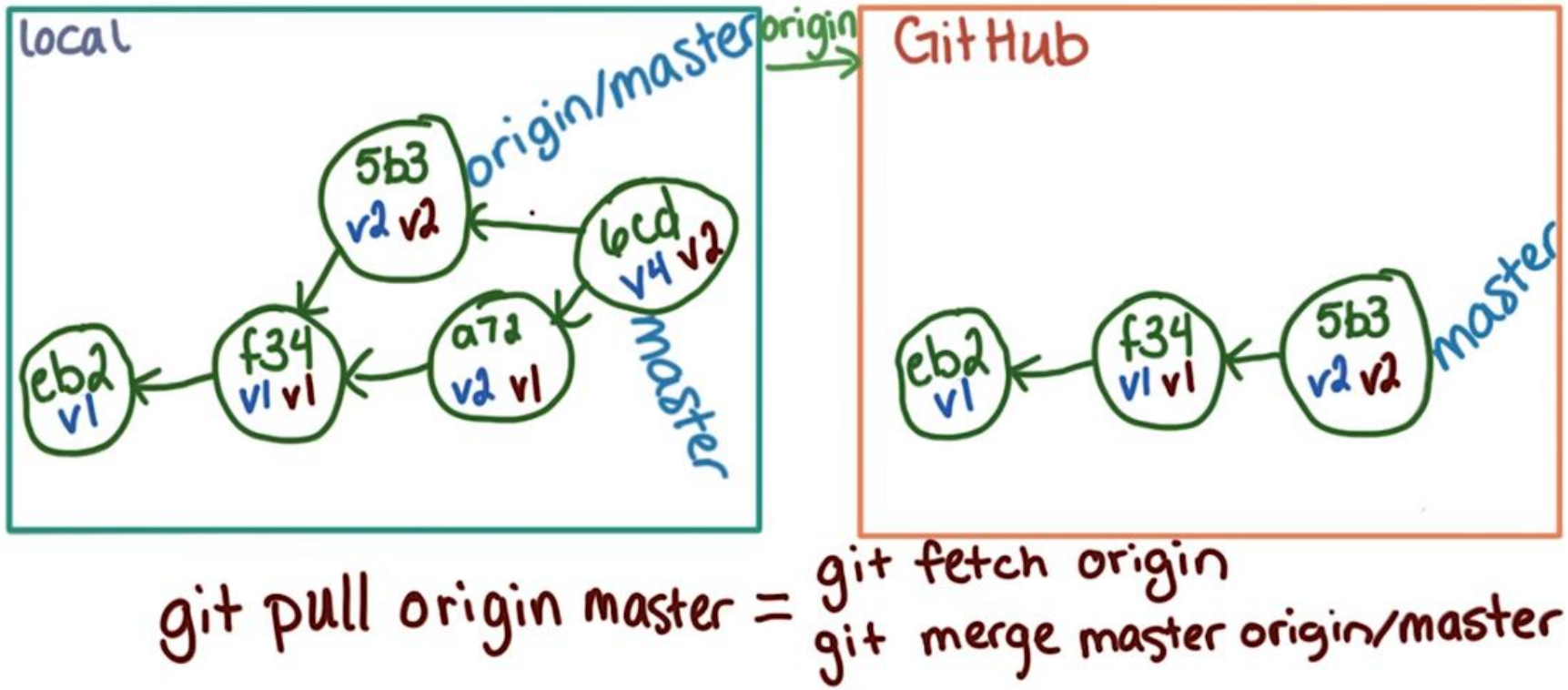
Importer les modifications sur la branche master locale.



La collaboration sur GitHub
Les conflits de collaboration

- ***git merge master origin/master***

Fusionner les branches master local et remote importée.



La collaboration sur GitHub
Fork d'un repository

C'est quoi un fork d'un repository ?

Une fonctionnalité sur le *GitHub* pour créer une copie d'un projet sur le *GitHub*.

La collaboration sur GitHub
Pull Request

C'est quoi une Pull Request ?

- Une fonctionnalité sur le *GitHub* pour demander au détenteur du *repository* de prendre en compte les modifications apportées sur le *repository* du fork.

Quelles actions pour accepter une Pull Request ?

- Code review.
- Fusion des branches.
- Gestion de conflits.

Plan

- Généralités sur les outils de gestion de versions
- La gestion de versions avec le Git
  - Les bases du Git
  - Utilisation des branches
  - Fusion des branches
- La collaboration sur GitHub
- Bonnes pratiques d'utilisation du Git et GitHub

Bonnes pratiques d'utilisation du Git et GitHub

- Valider (commit) pour la bonne raison.

Exemples

Bug fixé, une nouvelle fonctionnalité, une nouvelle langue, nouvel écran, etc.

- Valider (commit) fréquemment.
- Écrire un message commit descriptif.

Exemple. *https://udacity.github.io/git-styleguide/index.html*.

Bonnes pratiques d'utilisation du Git et GitHub

- Tester avant de valider (commit).
- Analyser le code avant de valider (commit).
- Analyser le code (code review) avant d'accepter une Pull Request.
- Utiliser les outils de collaboration pour minimiser les conflits de fusion.

Exemples. *Slack*, *Mattermost, Rocket.Chat*, etc.

Références

1. Chacon, Scott, and Ben Straub. Pro git. Apress, 2014
2. https://en.wikipedia.org/wiki/Distributed_version_control
3. https://git-scm.com/
4. https://en.wikipedia.org/wiki/Git
5. https://en.wikipedia.org/wiki/GitHub