Formation Git

Raphaël FROMENTIN

>~/ATILLA

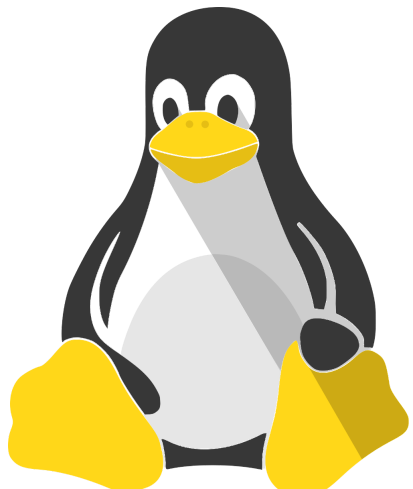
Sommaire

- Histoire de Git
- Avantages
- Fonctionnement
- Commandes de base
- Live coding

Un peu d’histoire

Les origines de Git

Linus Torvalds



Linux

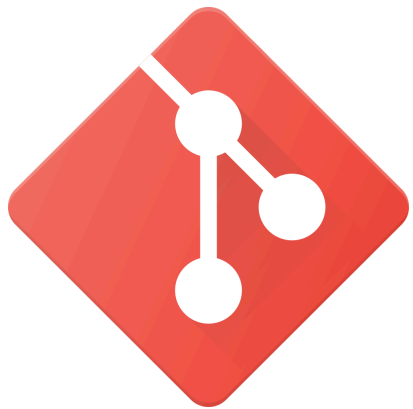
Situation de l’époque

Envoi de bouts de code par mail

- Beaucoup d’allers-retours manuels
- Intégration des changements dans le code principal assez douloureux
- Gérer plusieurs personnes qui développent en parallèle sans tout casser est difficile

La solution

Git !



"Git" veut dire "Connard" en argo britannique

La solution

Git !

Avantages:

- Chacun peut travailler en parallèle
- Chacun peut suivre ce que font les autres

Même en solo:

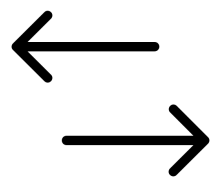
- Historique des changements
- Facile de coder sur plusieurs appareils
- Facile de tester des approches et pouvoir rollback à tout moment

Fonctionnement

Version Control System (VCS)

Échange entre plusieurs dépôts de code

Dépôt local



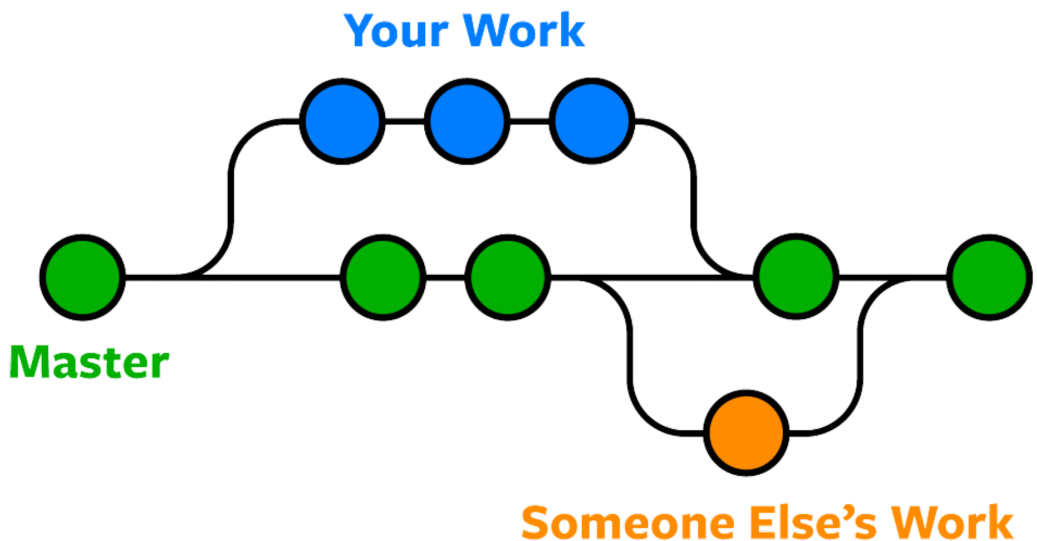
Dépôt distant/remote
(e.g sur GitHub, GitLab, ForgeJo...)

Fonctionnement

Version Control System (VCS)

Un dépôt contient plusieurs branches

C'est à dire plusieurs historiques alternatifs



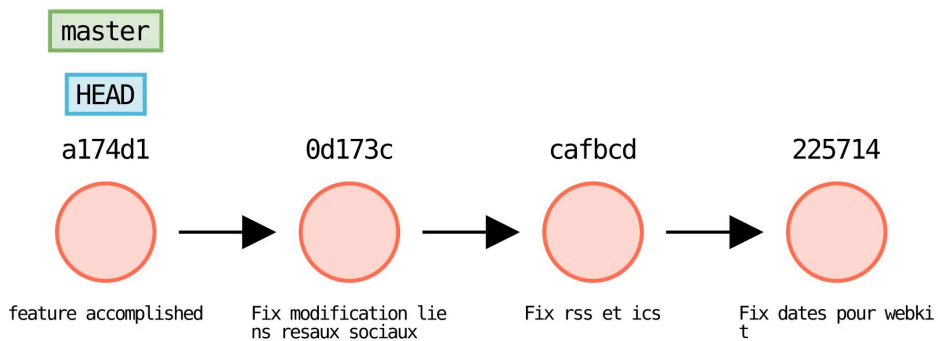
Fonctionnement

Version Control System (VCS)

Une branche est une suite de commits

Un commit est un groupe de changements annotés d'un message et d'un auteur.

Exemple à partir de l'historique de Tekiens.net :



Petit kiff personnel avec Git Story

Cas pratique

Toujours avec Tekiens.net

Un bug dans Tekiens !

```
if (Array.isArray(newObject[field])) {
    const tempRes = arrayDifference(originalObject[field], newObject[field]);
    if (tempRes.length > 0) res[field] = tempRes;
} else if (isObject(newObject[field])) {
    //Check if there's a difference
    const tempRes = objectDifference(originalObject[field], newArray[field]);
    if (Object.keys(tempRes) > 0) res[field] = tempRes;
} else if (newObject[field] ≠ originalObject[field]) {
    res[field] = newObject[field];
}
```

Cas pratique

Toujours avec Tekiens.net

Un bug dans Tekiens !

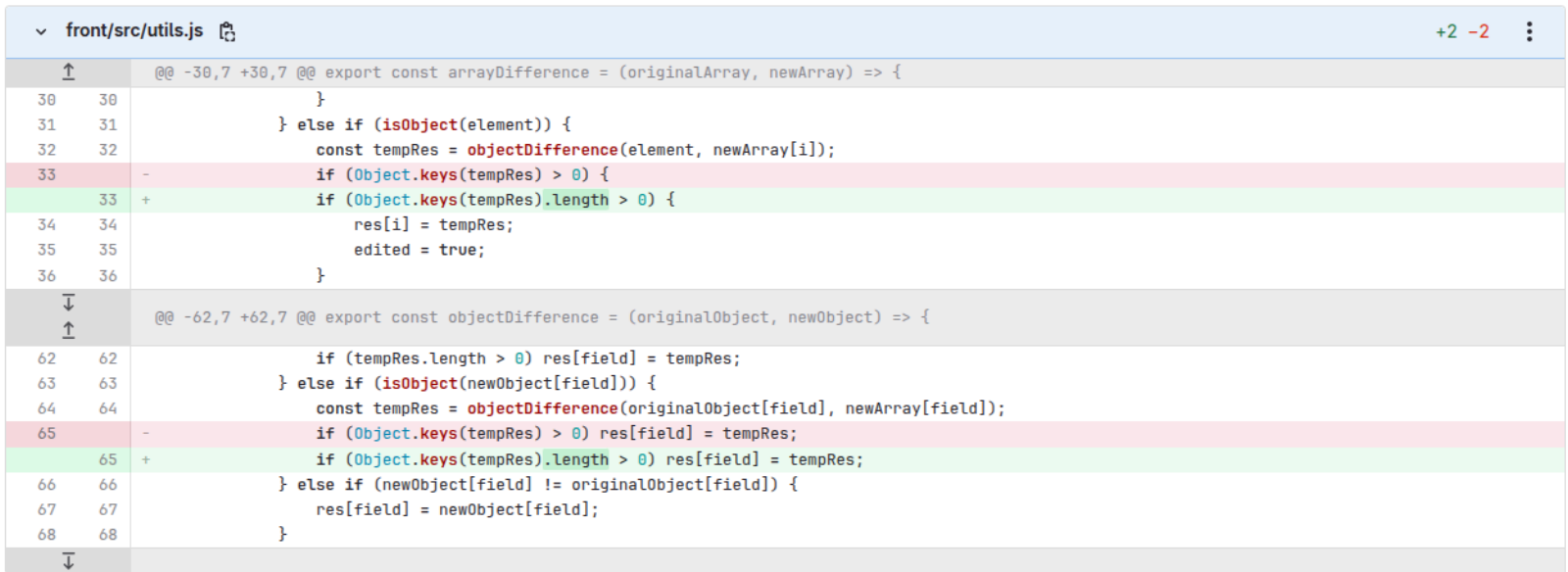
```
if (Array.isArray(newObject[field])) {
    const tempRes = arrayDifference(originalObject[field], newObject[field]);
    if (tempRes.length > 0) res[field] = tempRes;
} else if (isObject(newObject[field])) {
    //Check if there's a difference
    const tempRes = objectDifference(originalObject[field], newArray[field]);
    if (Object.keys(tempRes) > 0) res[field] = tempRes;
} else if (newObject[field] ≠ originalObject[field]) {
    res[field] = newObject[field];
}
```

On veut vérifier si `tempRes` n'est pas vide ⇒ `Object.keys(tempRes).length > 0`

Cas pratique

Toujours avec Tekiens.net

Étape 1 : faire les changements



front/src/utils.js +2 -2

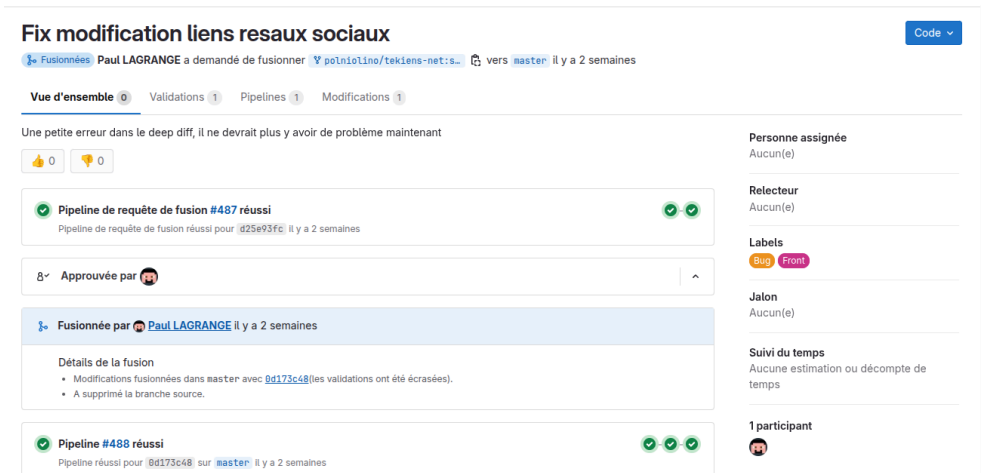
```
@@ -30,7 +30,7 @@ export const arrayDifference = (originalArray, newArray) => {
30 30                 }
31 31             } else if (isObject(element)) {
32 32                 const tempRes = objectDifference(element, newArray[i]);
33    -               if (Object.keys(tempRes) > 0) {
   33 +               if (Object.keys(tempRes).length > 0) {
34 34                     res[i] = tempRes;
35 35                     edited = true;
36 36                 }
@@ -62,7 +62,7 @@ export const objectDifference = (originalObject, newObject) => {
62 62                 if (tempRes.length > 0) res[field] = tempRes;
63 63             } else if (isObject(newObject[field])) {
64 64                 const tempRes = objectDifference(originalObject[field], newArray[field]);
65    -               if (Object.keys(tempRes) > 0) res[field] = tempRes;
   65 +               if (Object.keys(tempRes).length > 0) res[field] = tempRes;
66 66             } else if (newObject[field] != originalObject[field]) {
67 67                 res[field] = newObject[field];
68 68             }
```

Cas pratique

Toujours avec Tekiens.net

Étape 2 :

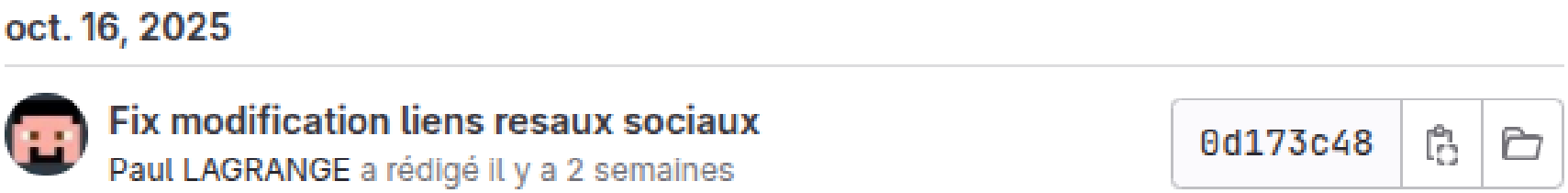
- Ouvrir une Merge Request (Gitlab) ou Pull Request (Github)
- La faire relire par quelqu'un (c'est mieux)



Cas pratique

Toujours avec Tekiens.net

Étape 3 : Fusionner les changements.



Et c'est tout !

Les commandes de base de Git

`< ... >` des paramètres obligatoires et `[ ... ]` des paramètres optionels

Cloner un dépôt distant (généralement la première commande que l'on fait) :

```
git clone <url>
```

Initialiser un dépôt local (pratique si vous avez déjà du contenu) :

```
git init
```

Lier son dépôt local à un dépôt distant (généralement ce qu'on fait après `git init`) :

```
git remote add <nom> <url>
```

Par convention, on nomme par défaut du dépôt distant `origin`.

Mettre à jour les informations des dépôts distants :

```
git fetch
```

Les commandes de base de Git

`< ... >` des paramètres obligatoires et `[ ... ]` des paramètres optionels

Lister les branches :

```
git branch [--show-current] [--all]
```

Changer de branche :

```
git switch <branch>
```

Afficher l'historique :

```
git log
```

Récupérer les changements :

```
git pull [--rebase]
```

En local, privilégiez avec `--rebase` pour éviter de polluer votre historique avec un commit de merge (ce qui se passe sans `--rebase`).

Les commandes de base de Git

`< ... >` des paramètres obligatoires et `[ ... ]` des paramètres optionels

Stage ses changements (pour les inclure dans le prochain commit) :

```
git add <fichier1> <fichier2> ...
```

On peut passer des dossiers directement comme `.` pour inclure tout le dossier actuel.

Créer un commit :

```
git commit [-m <message>]
```

Si `-m` n'est pas précisé, Git vous demandera d'entrer un message, souvent via l'éditeur Vim par défaut (`:q` pour quitter/confirmer).

Push les nouveaux commits :

```
git push [--force-with-lease]
```

Privilégiez le push sans force. Si vous devez le faire, préférez `--force-with-lease` à `--force` et évitez de le faire sur votre branche principale aka `main` ou `master`.

Vous n’aimez pas le terminal ?

Des alternatives existent

Clients graphiques :

- GitKraken Desktop
- GitHub Desktop
- ...

Intégration dans la plupart des éditeurs de code et IDEs :

- IDEs JetBrains (Intellij IDEA, PyCharm, PHPStorm...)
- VSCode
- ...

Mais savoir se débrouiller avec le terminal permet d’utiliser Git quelque soit les autres outils à disposition.

Exemple pratique

Faire une Pull Request sur un dépôt GitHub

https://github.com/lltotore/git-course-example

Merci de m'avoir écouté !

Les slides sont sur GitLab:
https://gitlab.atilla.org/raphael.fromentin1/formation-git-2025/

Et en PDF sur https://slides.atilla.org/training-git-2025-2026.pdf