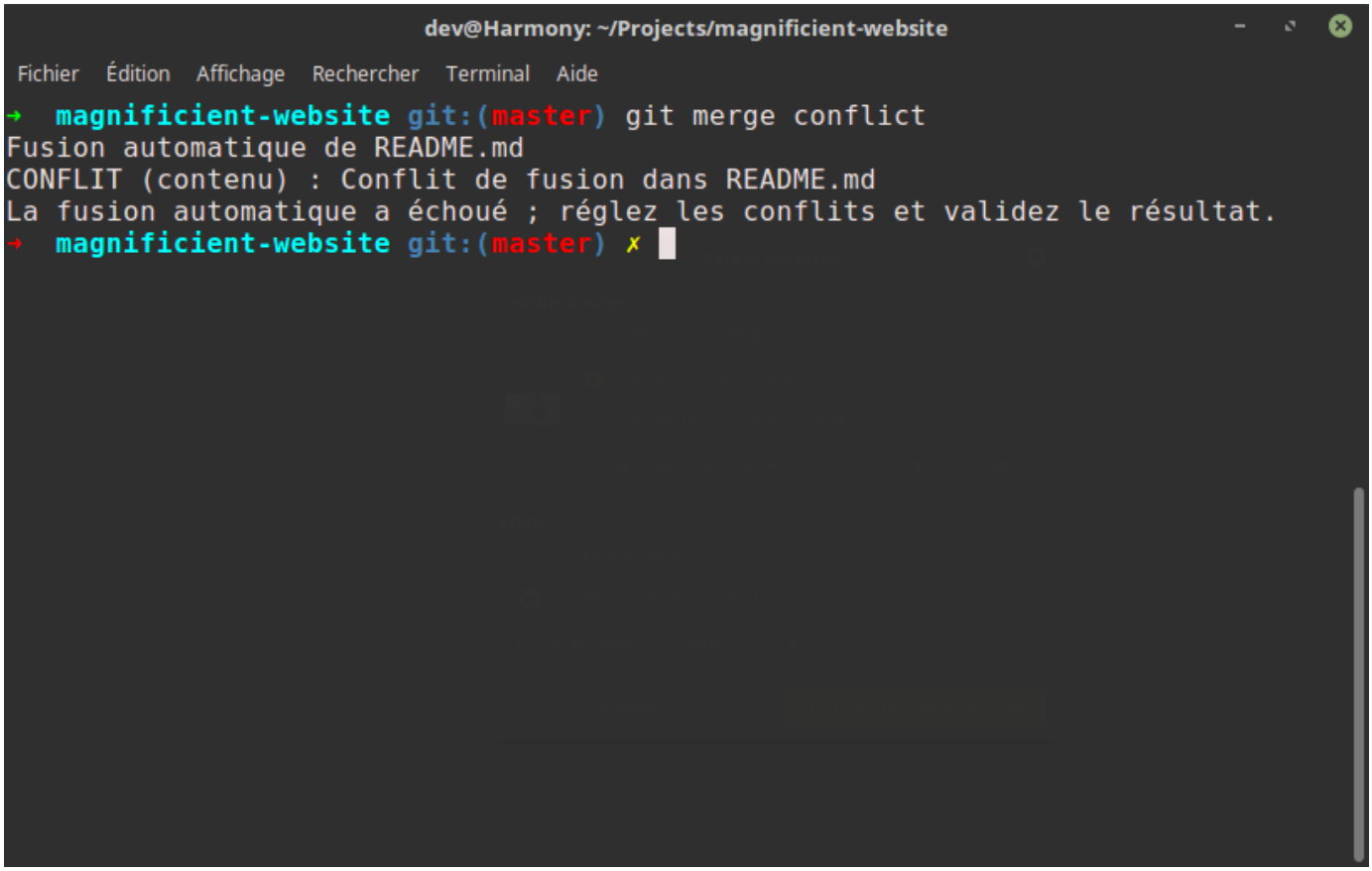
Gérer un conflit git

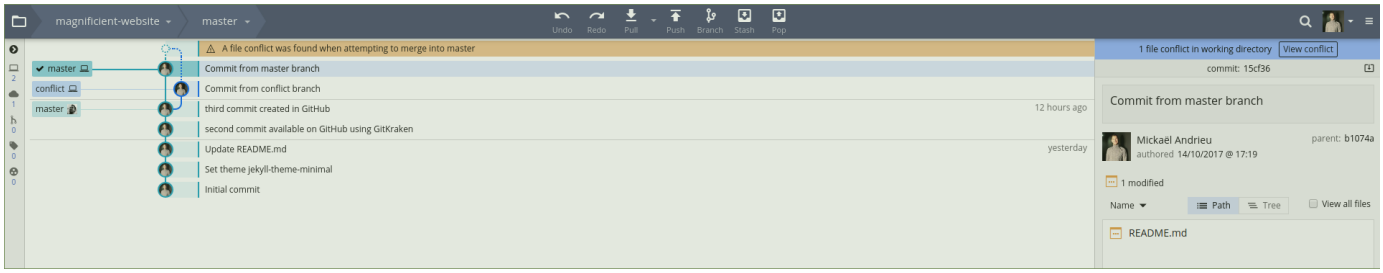
Il arrivera quand on travaille en équipe que vous soyez en situation où plusieurs membres travaillent sur les mêmes fichiers, voir sur les mêmes ligne d'une fichier. Lors du merge de la branche de fonctionnalité dans la branche master, Git ne saura pas quoi et il faudra donc prendre décider ce qu'il convient de faire.

Création d'un conflit

> *Vous pouvez effectuer cette partie avec le terminal ou GitKraken*

- Ouvrir le terminal et accéder au dossier "magnificient-website" pour créer une nouvelle branche git appelée "conflict" (`git checkout -b conflict` pour rappel).
- Sur cette branche "conflict", ouvrir le fichier "README.md" et changer la première ligne par "# Conflict commit from *conflict* branch".
- Ajouter le fichier.
- Commiter avec le message "Commit from conflict branch"
- Se déplacer sur la branche "master" (`git checkout master` pour rappel).
- Ouvrir le fichier "README.md" et changer la première ligne par "# Conflict commit from *master* branch".
- Ajouter le fichier.
- Commiter avec le message "Commit from master branch".
- Merger la branche "conflict" dans master: `git merge conflict` :





Cet exemple est volontairement simpliste: Git ne sait tout simplement pas quelle “première” ligne il doit conserver!

On a plusieurs choix:

- Conserver le choix de la branche “HEAD”, c’est à dire celle où on se trouve (“master” dans ce cas);
- Conserver le choix de la branche “conflict” qu’on souhaite merger/ajouter dans “master”;
- Conserver les 2;
- Tout changer et faire un choix complètement différent (râre, mais possible);

> *Ce choix appartient au développeur/contributeur qui peut et devrait communiquer avec les autres personnes qui ont effectué des changements s’il n’est pas en capacité de prendre une décision.*

```
1  <<<<<<< HEAD (Modification actuelle)
2  # Conflict commit from *master* branch
3  ||||||| third commit created in GitHub
4  # This is an update from local with GitKraken
5  =======
6  # Conflict commit from *conflit* branch
7  >>>>>>> conflict (Modification entrante)
8  ## Edité à partir de GitHub
```

Dans note cas, nous allons conserver la modification qui arrive de la branche “conflit”.

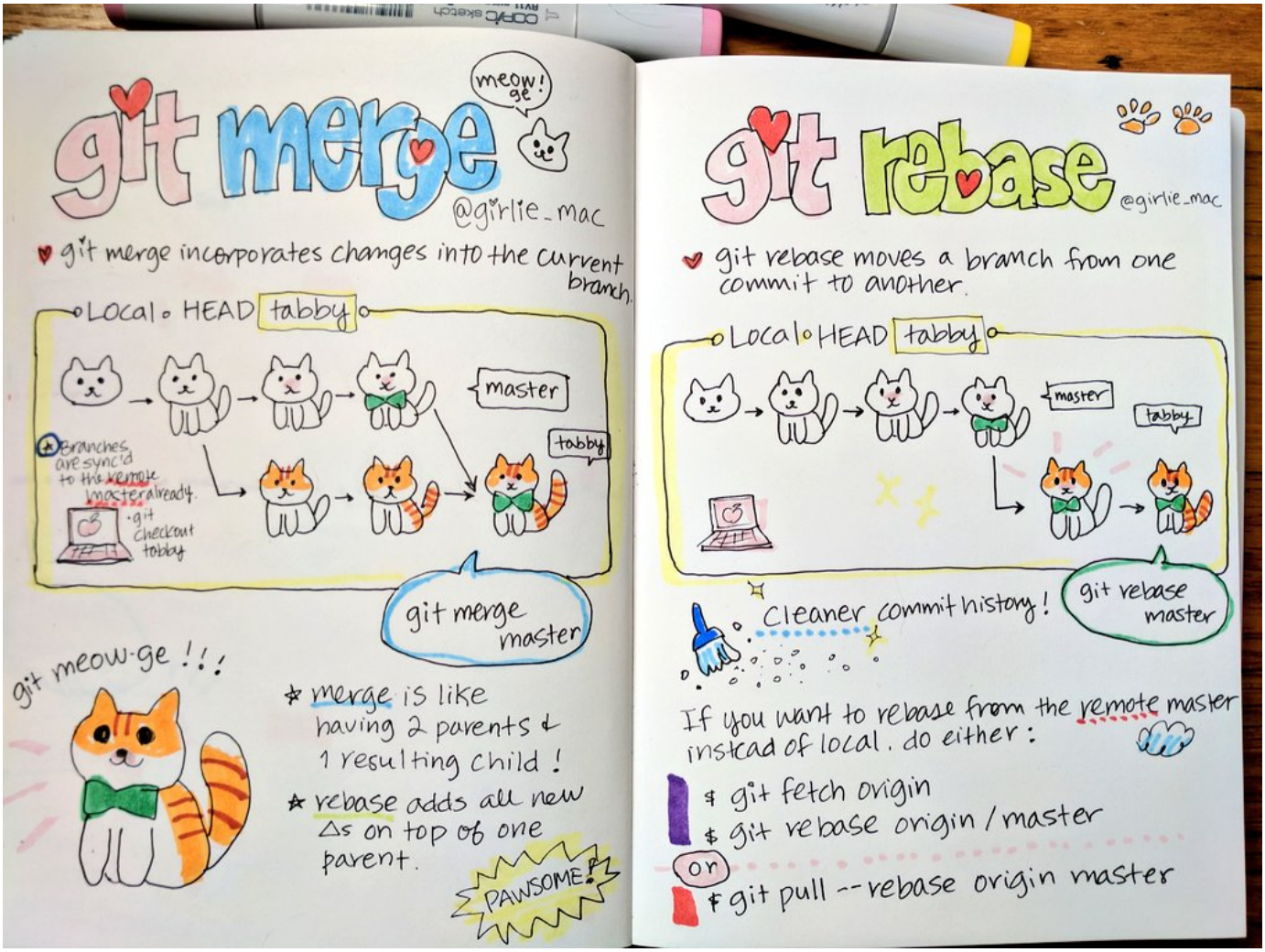
- Ouvrir le fichier "README.md" et ne conserver que les lignes 6 & 8.
- Ajouter le fichier.
- Exécuter la commande `git commit` : un commit de résolution de conflit a été créé.

La différence entre git merge et git rebase

Lorsque l’on souhaite rappatrier des fonctionnalités issues d’une autre branche git, nous avons 2 choix:

- Effectuer un `git merge` ;
- Effectuer un `git rebase` ;

Il est assez difficile d’expliquer avec des mots la différence entre les 2. Voici un petit schéma qui vous aidera à comprendre la différence.



Quand on merge, la gestion des conflits se fait lors du merge alors que le rebase permet d'incorporer les modifications "commit par commit". C'est donc plus fin, plus propre, mais également un peu plus compliqué à gérer quand on débute.

Si on avait voulu gérer le conflit de la façon la plus propre possible, voici ce qu'il aurait fallu faire:

- Aller sur la branche "conflict" et exécuter la commande `git rebase master`.
- Régler le conflit.
- Retourner sur la branche "master" et merger "conflict" dedans (`git merge conflict`).