

Logiciel de gestion de version

- Facilite le travail en équipe
- Evite des pertes de données
- Permet de capturer des instants du projets (fonctionnel ou non)

Sauvegarde 1
du projet

Sauvegarde 2
du projet

Etc

Git ?

- Gestion décentralisée (travail à son rythme)
- Echange des travaux à travers une plateforme (**GitHub**, Gitlab, Gitea...)

Comment utiliser Git ?

Installation de git :

pacman -S git
apt-get install git

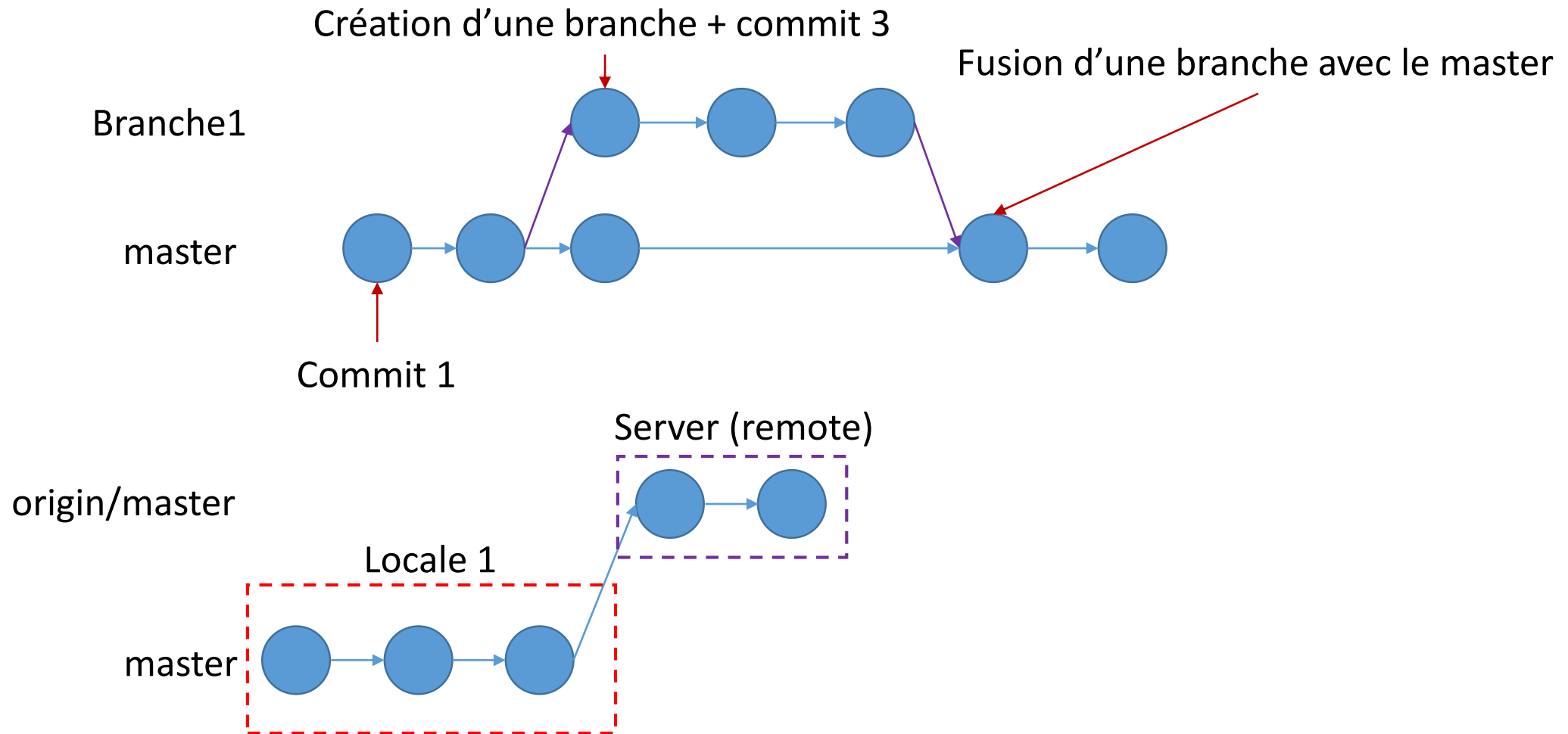
A travers :

- Une interface graphique tierce (gitkraken, tortoise-git)
- Le terminal (utilisation de gitbash sur Windows)

Utilisation de commandes du type :

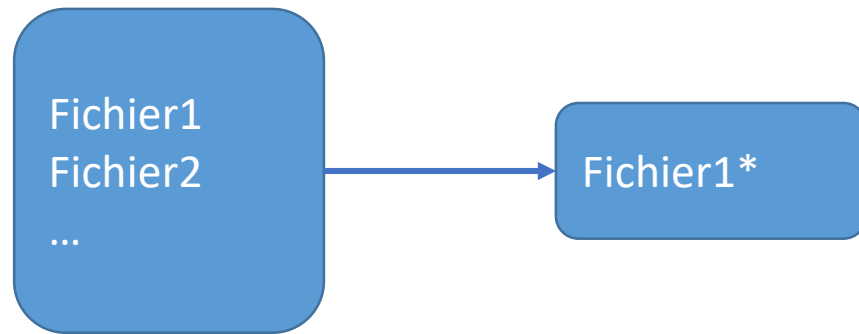
git <nom de la commande> <paramètres>

A quoi ça ressemble ?



Le « commit »

Création d'une photo du projet



Ne garde en mémoire que les modifications d'un commit à l'autre

Commandes :

git add Fichier1 ... (ou **git add --all**) (Permet de considérer les fichiers mis en paramètre pour le commit)

git status (vérifier les fichiers ajoutés)

git commit -m « Nom du commit visible par tous » (décrit les changements réalisés pour ce commit)

Le « checkout »

Permet de se déplacer d'un commit à un autre



git log (permet de voir les noms des commits)

git checkout nomDuCommit

Certain commit possèdent des noms spéciaux :

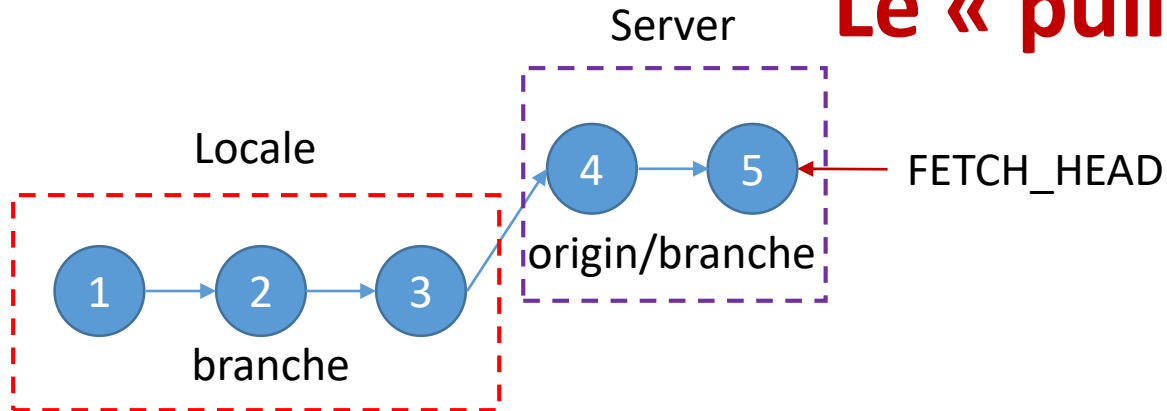
git checkout nomBranche (déplace sur le dernier commit d'une branche)

git checkout HEAD (déplace sur la tête de la branche actuelle)

git checkout FETCH_HEAD (déplace à la tête du dernier fetch)

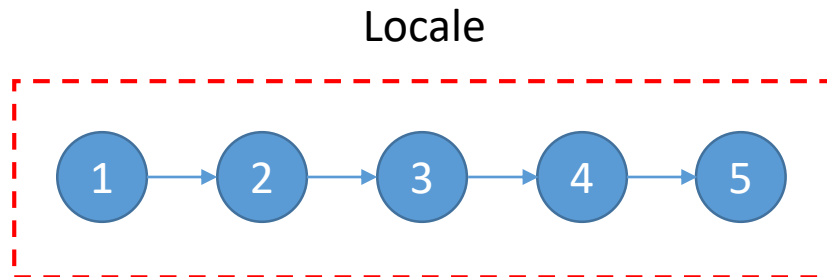
git checkout tag (déplace sur le commit possédant le tag associé)

Le « pull »



Récupération des commits manquant :

git pull (recherche les commits manquant et les fusionnent avec le projet)



Attention possibilité de conflits

Méthode plus sûr :

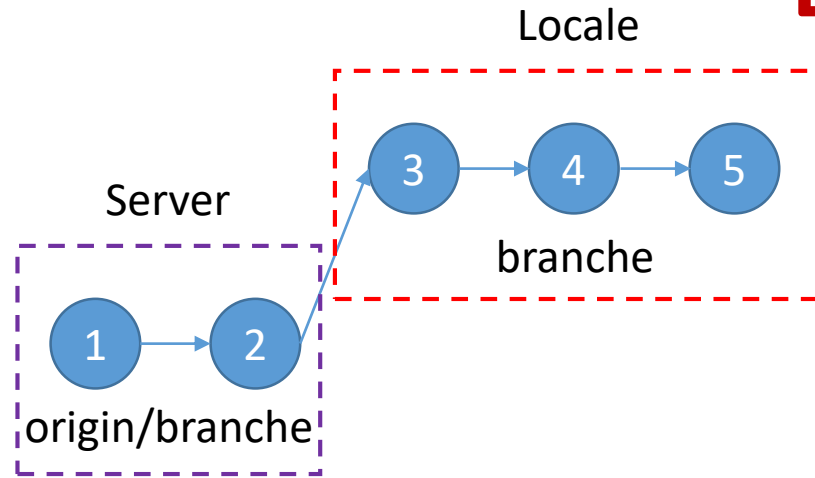
git fetch (synchronisation avec le serveur)

git log FETCH_HEAD (permet de voir les commits sur le serveur)

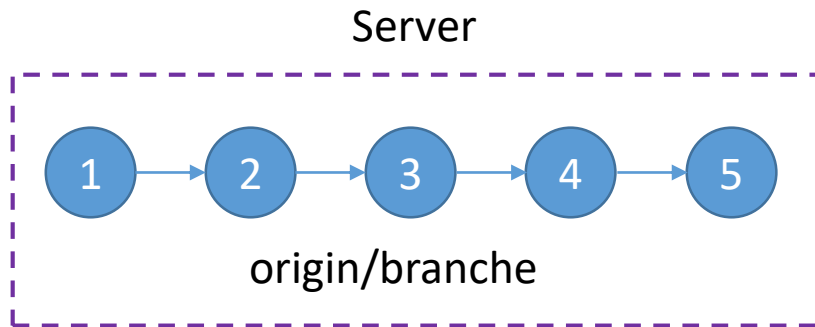
git diff --name-only FETCH_HEAD (permet de voir rapidement s'il n'y a pas de conflit)

git merge FETCH_HEAD (récupère les données)

Le « push »



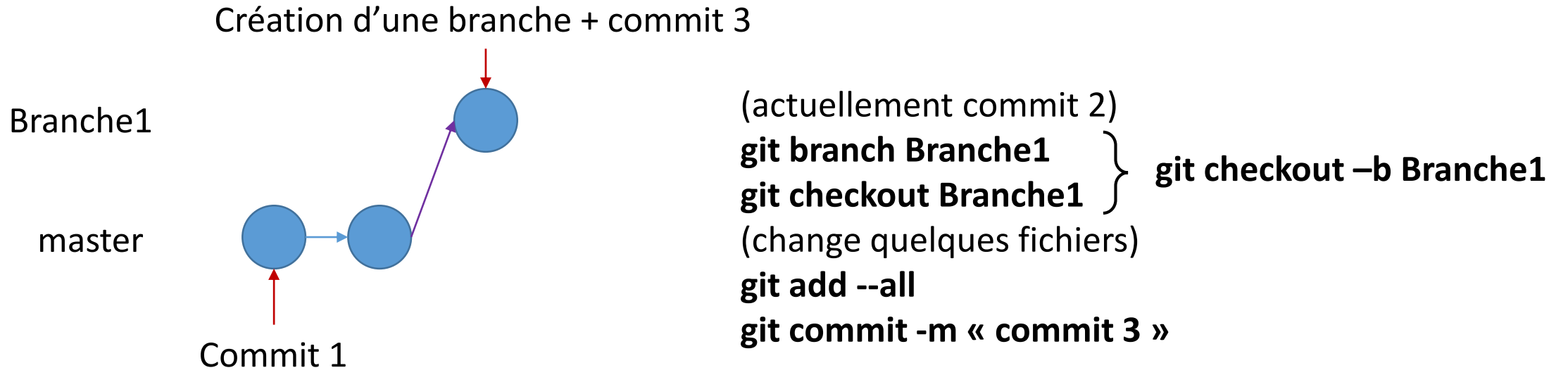
Envoie des commits manquant sur la plateforme :
git push



Attention avant d'envoyer les commits, il faut mettre à jour le répertoire local (voir « pull »)

Les branches

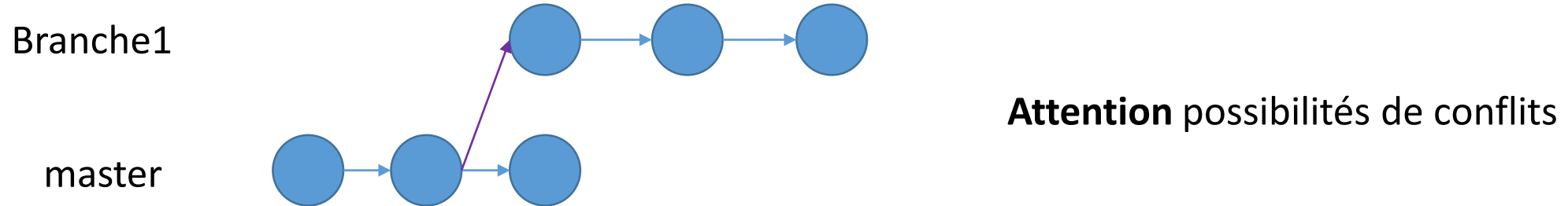
Permet de créer une branche de travail



Branches existantes et branche actuelle :
git branch -a

Le « merge »

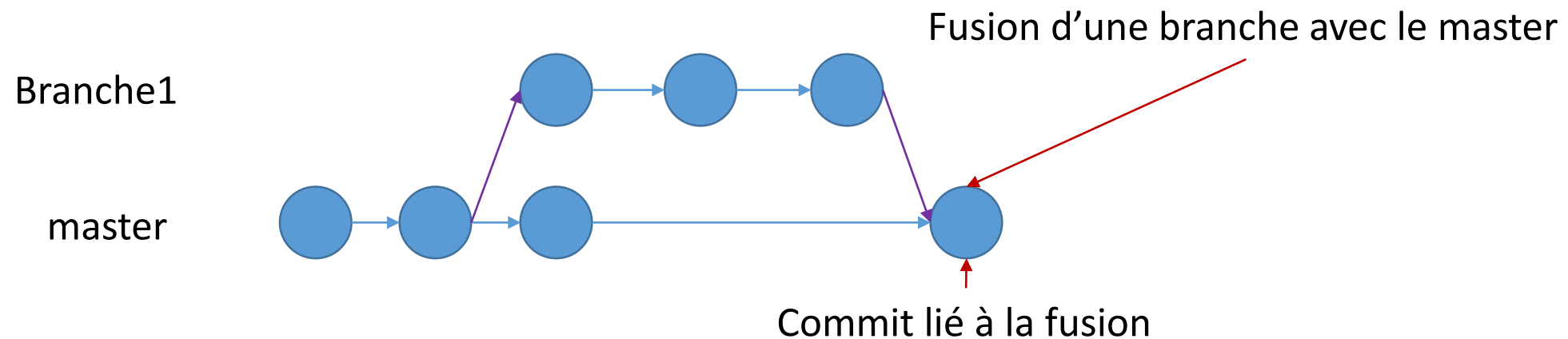
Permet de fusionner les changements d'un commit avec un autre



git status (voir la branche actuelle)

git checkout master

git merge Branche1 (pour ramener les informations de Branche1 dans master)

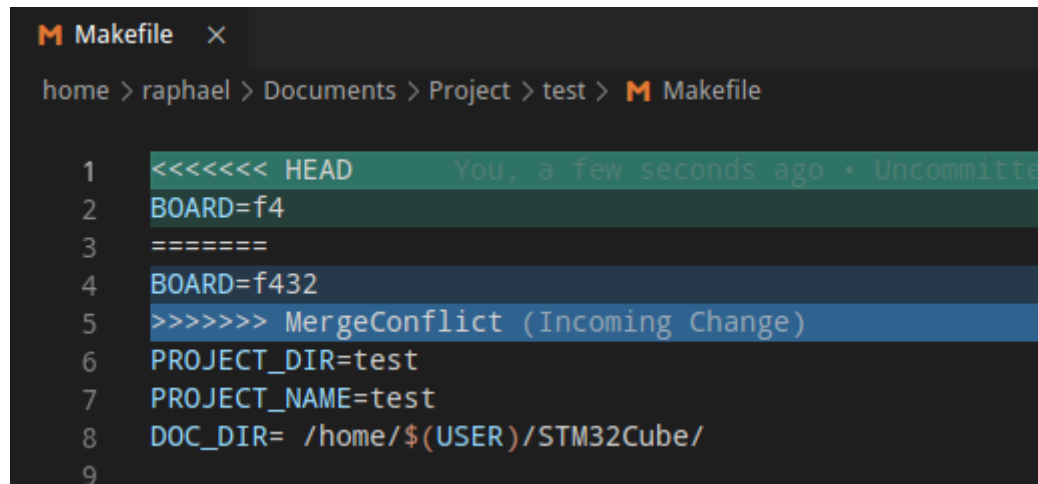


Les conflits

La fusion de commit peut entraîner des conflits :
Modification différente sur le même fichier

```
raphael@raphael | ~/Documents/Project/test | master | git merge MergeConflict
Auto-merging Makefile
CONFLICT (content): Merge conflict in Makefile
Automatic merge failed; fix conflicts and then commit the result.
```

Traiter le conflit à travers un éditeur de texte

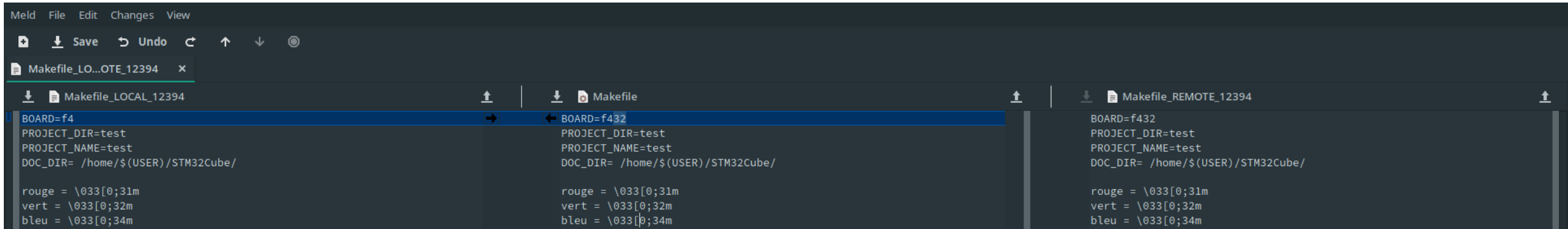


```
M Makefile x
home > raphael > Documents > Project > test > M Makefile

1 <<<<<< HEAD      You, a few seconds ago • Uncommitte
2 BOARD=f4
3 =====
4 BOARD=f432
5 >>>>>> MergeConflict (Incoming Change)
6 PROJECT_DIR=test
7 PROJECT_NAME=test
8 DOC_DIR= /home/$(USER)/STM32Cube/
9
```

Les conflits

Traiter le conflit à travers un outil de fusion (merge tool)



Exemple d'outil : meld, kdiff3

Installation de meld :

pacman -S meld

apt-get install meld

Mise en place avec git :

git config --global merge.tool meld

Mise en pratique

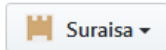
Création d'un répertoire sur GitHub

Create a new repository

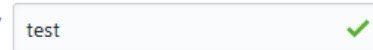
A repository contains all project files, including the revision history. Already have a project repository elsewhere?

[Import a repository.](#)

Owner

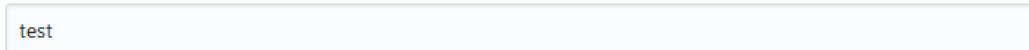


Repository name *



Great repository names are short and memorable. Need inspiration? How about [cuddly-journey?](#)

Description (optional)



Public

Anyone can see this repository. You choose who can commit.



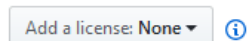
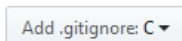
Private

You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

☒ Initialize this repository with a README

This will let you immediately clone the repository to your computer.



Create repository

Le « gitignore » permet d'ignorer des fichiers suivant un pattern (fichiers de build et autres)

Mise en pratique

Cloner le répertoire en local

Suraisa / AssemblerMIPS

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Actions Projects 0 Wiki Security Insights Settings

No description, website, or topics provided. [Manage topics](#)

91 commits 1 branch 0 packages 0 releases 2 contributors

Branch: master New pull request

Create new file Upload files Find file Clone or download

Clone with HTTPS ⓘ Use SSH

Use Git or checkout with SVN using the web URL.

<https://github.com/Suraisa/AssemblerMIPS>

Open in Desktop Download ZIP

File	Description
doc	Add and fix the relocation table, add some possibility in the operand
files	Fix errors .word
inc	Refactoring
lib	Add instruction type R

git clone Adresse (crée un dossier connecté à l'adresse possédant le nom du répertoire sur git)

Travail en local ou mise en place en local :
git init (à la racine du projet)

Mise en pratique

Configurer son git

`git config --global user.name « nom »` (permet de spécifier qui travail)

`git config --global user.email adresse@example.com`

Ajout d'alias pour simplifier les commandes :

`st = status`

`df = diff`

`co = checkout`

`ci = commit`

`cim = "!f() { git commit -m \"$1\"; }; f"`

`amend = commit --amend`

`br = branch`

`bra = branch -a`

`diffstat = df --stat`

`undo = "!f() { for i in \"$@\"; do git checkout -- \"$i\"; done }; f"`

`revertall = reset --hard`

`fe = fetch -p`

`pe = "!f() { git fetch -p && git pull; }; f"`

`revert = "!f() { git checkout -- \"$1\"; }; f"`

`cob = "!f() { git checkout -t \"remotes/$1\"; }; f"`

Aller dans le fichier de configuration de git dans le Home et ajouté la balise suivante :

`[alias]`

`alias1 = commande`