CC BY SA

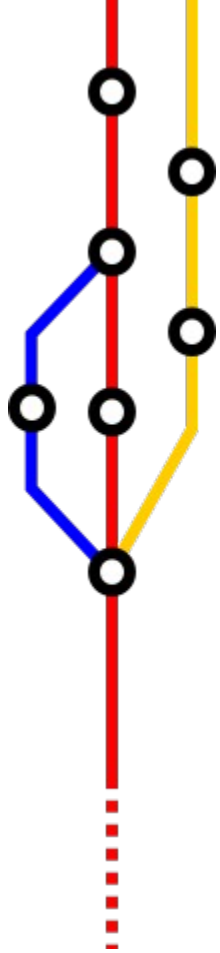
Club Informatique

How to Git Gud

Par Arthur Desuert

CC BY SA

Sommaire



- Introduction
- Bases de Git
- Travail d’équipe avec Git
- Les outils graphiques
- Bonus

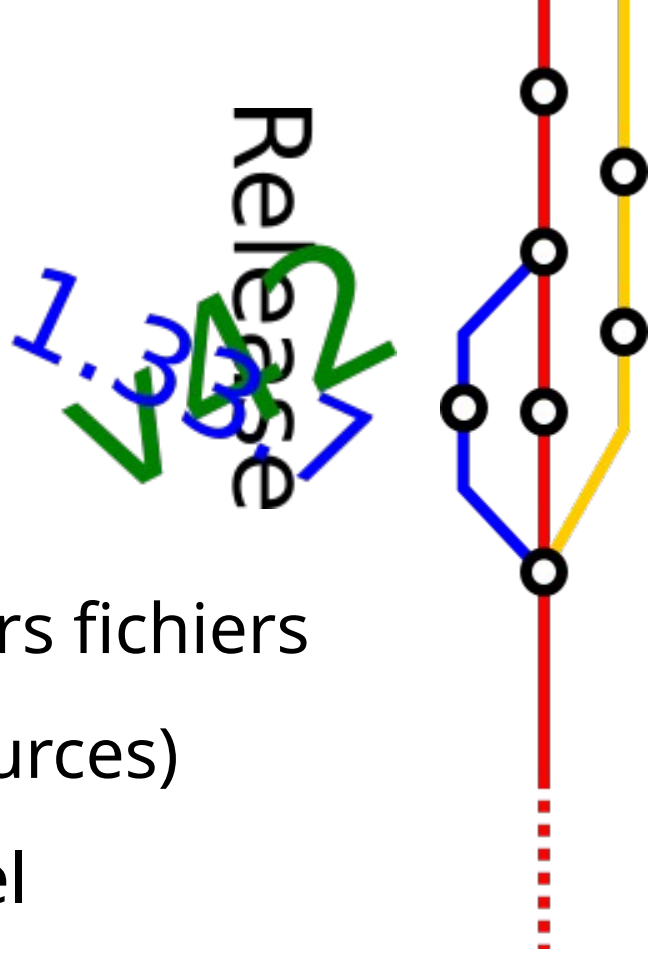
CC BY SA

Introduction

CC BY SA

Introduction

Contrôle de version



Le contrôle de version, c'est quoi ?

- Gestion des versions d'un ou plusieurs fichiers
- Principalement des fichiers texte (sources)
- Très utilisé en développement logiciel

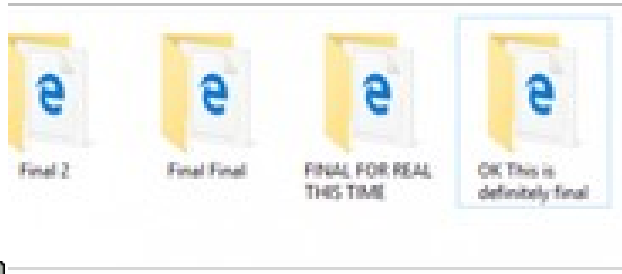
→ Souvent très complexe

AIDE LOGICIELLE REQUISE

CC BY SA

Introduction

Contrôle de version



CC BY SA

Introduction
Git

Le gestionnaire de versions Git :

- Créé en 2005 par Linus Torvalds
- Gratuit et open-source
- Assez répandu et apprécié

(cf https://insights.stackoverflow.com/survey/2018)

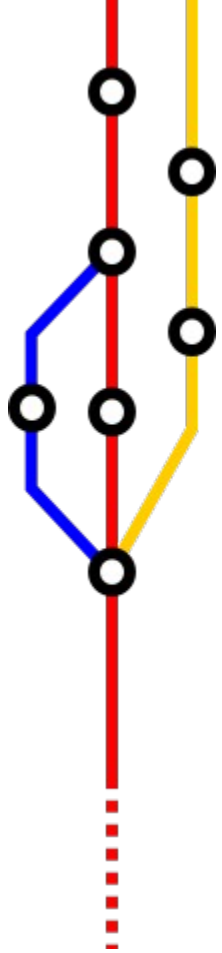
CC BY SA

Club Informatique

Les bases de Git

cc BY SA

Installation



Pour Debian et dérivés :

 # apt install (-y) git

Pour Fedora et dérivés :

 # dnf install git

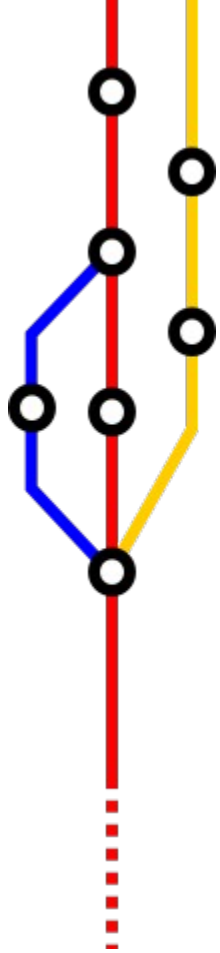
Pour Archlinux et dérivés :

 # pacman -S git

CC BY SA

Les bases de Git

Création d'un utilisateur



Déclarer son 'identité' à Git :

$ git config (--global) user.name "<pseudo_git>"

$ git config (--global) user.email "<email_git>"

/!\ Ces infos sont incluses dans les commits /!

CC BY SA

Le premier dépôt (local)

Au commencement, il y avait :

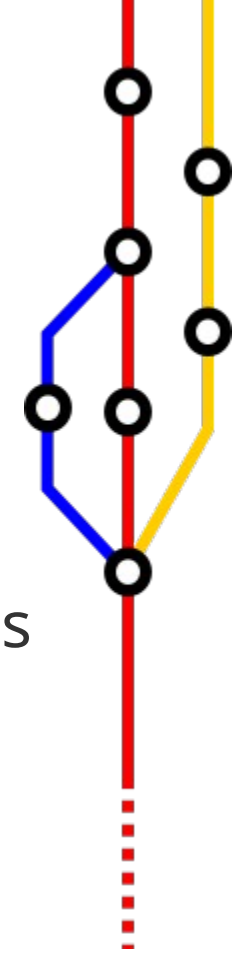
$ git init

Init. d'un dépôt dans le répertoire courant

→ Création d'un dossier .git

CC BY SA

Fichiers de départ



- Le **README** :
  - Permet d'expliquer le contenu du dépôt, donner les directives de config, etc.
  - Peut être rédigé en texte simple, Markdown,...

- Le **.gitignore** :
  - Permet d'ignorer automatiquement certains fichiers.
  - Basé sur du pattern matching
  - Très utile pour ne pas polluer son dépôt

CC BY SA

Les opérations de base

Puis vinrent les premiers fichiers :

```
[desuerta:/tmp/my_git] $ ls
file1  file2
```

Pour voir ce qu'en dit Git :

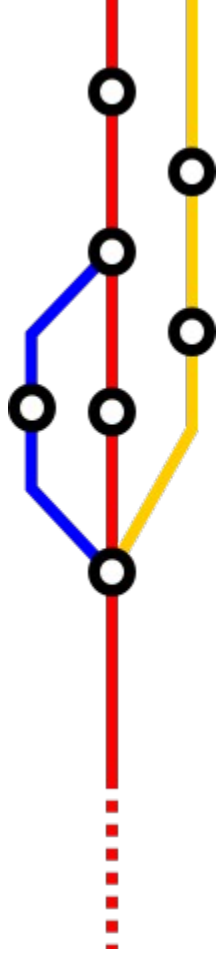
$ git status

```
[desuerta:/tmp/my_git] $ git status
Sur la branche master

Validation initiale

Fichiers non suivis:
  (utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)

        file1
        file2

aucune modification ajoutée à la validation mais des fichiers non suivis sont p
```


CC BY SA

Club Informatique

Les opérations de base



2 étapes clés du 'workflow' Git :

1) Sélection des modifications :

$ git add <file1> <file2> ... <fileN>

2) Validation de la sélection (le 'commit')

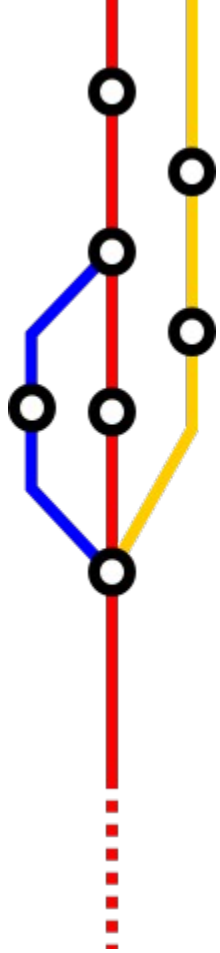
$ git commit (-m <message>)

Étape intermédiaire : vérifier le contenu de la sélection

$ git status

CC BY SA

Les opérations de base



Quelques commandes utiles :

- Visualiser l'historique :

$ git log (<file>)

- Voir les différences :

$ git diff (<hash_c1> <hash_c2>)

cc BY SA

Travail d'équipe avec Git

CC BY SA

Les 'hébergeurs' Git

Il existe des 'hébergeurs' publiques :

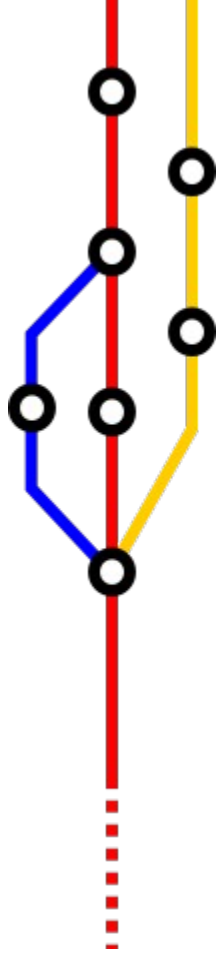
- Github (Micro$oft)
- Framagit (Framasoft)

Auto-hébergement possible !

- Gitlab
- Serveur Git natif

CC BY SA

Récupérer un dépôt existant



Clonage d'un dépôt avec Git :

$ git clone <url_dépôt>

Remarques :

- URL doit pointer vers 'racine' du dépôt
- Clonage (par défaut) de tout l'historique

CC BY SA

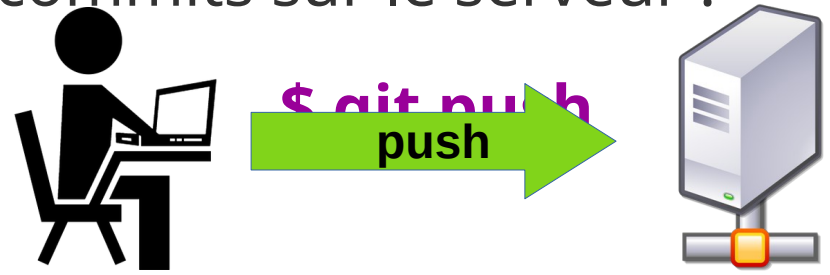
Travail avec un dépôt distant

Travail en local inchangé

Interactions avec le serveur distant :

- Mise à jour de son dépôt avec le distant :

- Partage des commits sur le serveur :



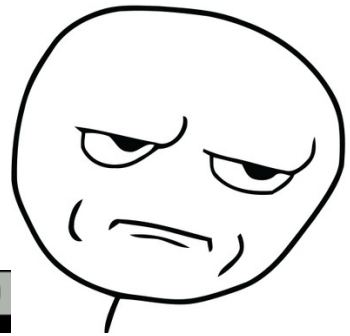
CC BY SA

Travail d'équipe avec Git
Gérer les conflits (1/2)

Lors d'un merge de branches, il peut y avoir des conflits : 2 versions différentes d'un fichier

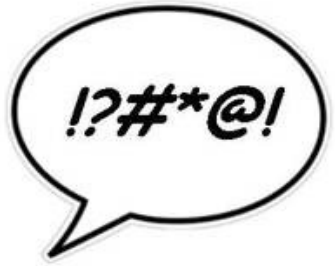
→ Cas d'un **pull** par exemple !

```
1& [desuerta:/tmp/hello-world] master(1)* ± git pull
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Dépaquetage des objets: 100% (3/3), fait.
Depuis https://github.com/Athuro83/hello-world
   c8fac73..52278ea  master     -> origin/master
Fusion automatique de new.txt
CONFLIT (contenu) : Conflit de fusion dans new.txt
La fusion automatique a échoué ; réglez les conflits et validez le résultat.
```



CC BY SA

Travail d'équipe avec Git
Gérer les conflits (2/2)



2 approches possibles :

- Abandonner le merge :

$ git merge --abort

- Résoudre le conflit :

1) 'Fusionner' manuellement les fichiers en conflit
2) Faire un commit

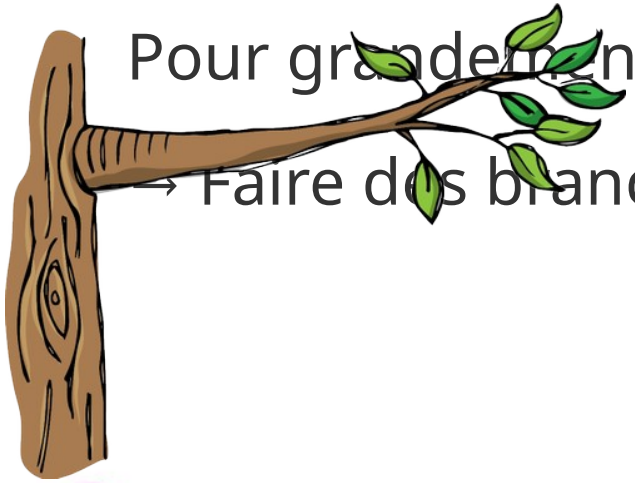
Outils : git diff, meld, kdiff3, votre éditeur favori,...

CC BY SA

Éviter les conflits

Pour éviter les conflits :

→ Ne pas travailler à plusieurs sur un même fichier

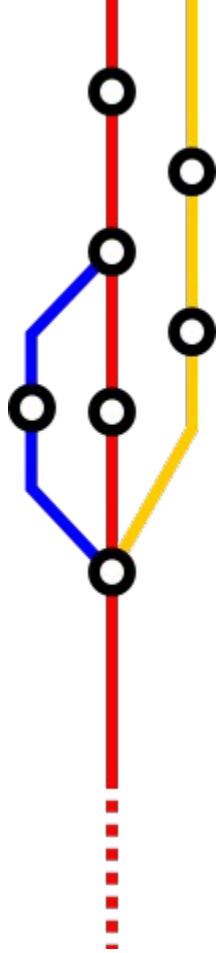


Pour grandement réduire les conflits :

→ Faire des branches !

CC BY SA

Les branches (1/2)



Pour créer une branche (et se placer dessus) :

$ git checkout -b <nom_branch>

Pour lister les branches existantes :

$ git branch

Pour passer d'une branche à l'autre :

$ git checkout <nom_branche>

CC BY SA

Les branches (2/2)

Pour « merge » deux branches ensembles :

1) Se placer sur la branche à conserver
2) Effectuer le « merge » :

$ git merge <nom_branche>

/!\ Conflits possibles /!

Pour supprimer une branche :

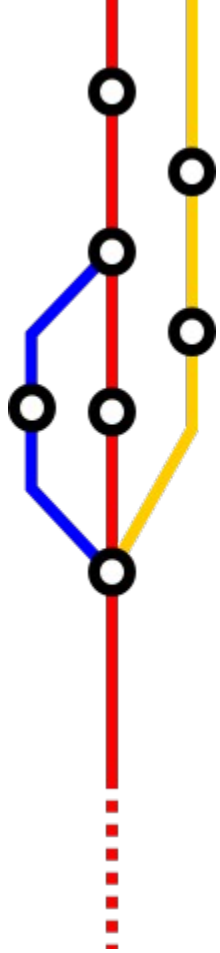
$ git branch -d <nom_branche>

CC BY SA

Les outils graphiques

cc BY SA

Les outils graphiques
C'est quoi ?



- Une interface graphique
- Intégré ou non à un environnement (IDE)
- Pour tous les goûts !

Quelques exemples :

git-gui, git-cola, GitKraken, Guitar

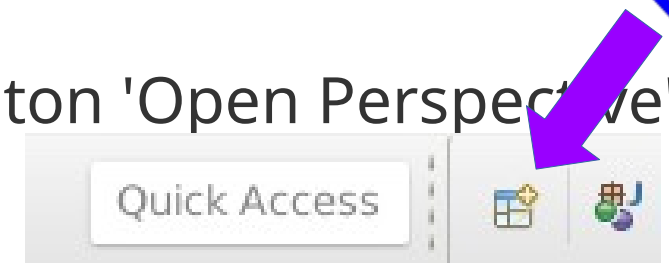
CC BY SA

Les outils graphiques

Focus : le client Git d'Eclipse

Première méthode : ouvrir la perspective Git

- Cliquez en haut à droite sur le bouton 'Open Perspective'
- Sélectionnez la perspective **Git**



- Changement de perspective avec les boutons en haut à droite.

CC BY SA

Les outils graphiques

Focus : le client Git d'Eclipse

[PRESENTATION LIVE]

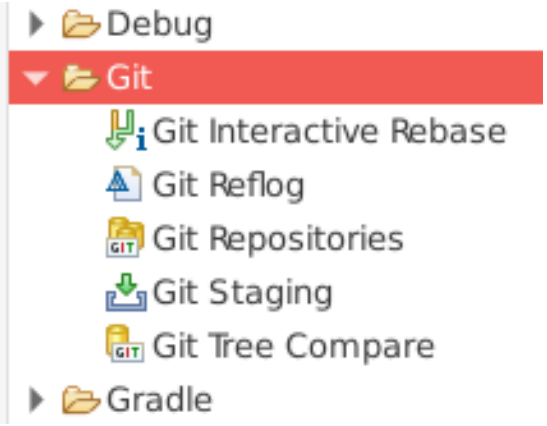
Page Wiki dédiée: https://wiki.eclipse.org/EGit/User_Guide

CC BY SA

Focus : le client Git d'Eclipse

Seconde méthode : ajouter seulement les fenêtres Git utiles

- Dans l'onglet 'Window > Show view > Other...'
- Dans le répertoire Git :



- Les fenêtres importantes :
  - Staging
  - Repositories
  - History

CC BY SA

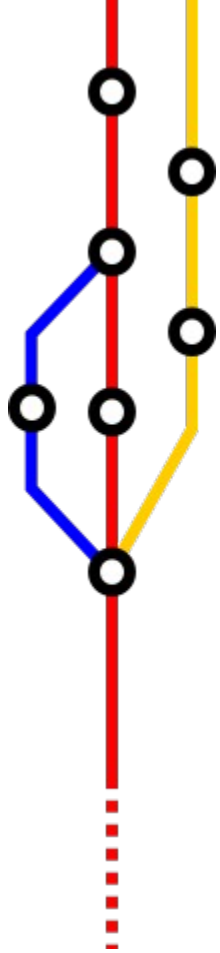
Club Informatique

Bonus !

CC BY SA

Bonus

Les alias Git



Similaire aux alias Linux :

$ git config (--global) alias.<alias> <commade>

Quelques alias pratiques :

$ git config alias.st status

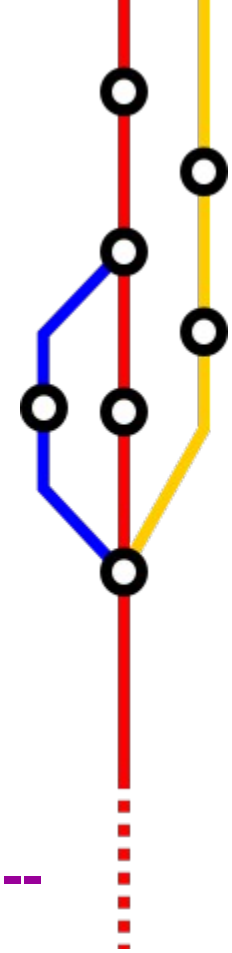
$ git config alias.ci commit

$ git config alias.co checkout

CC BY SA

Bonus

Les alias Git



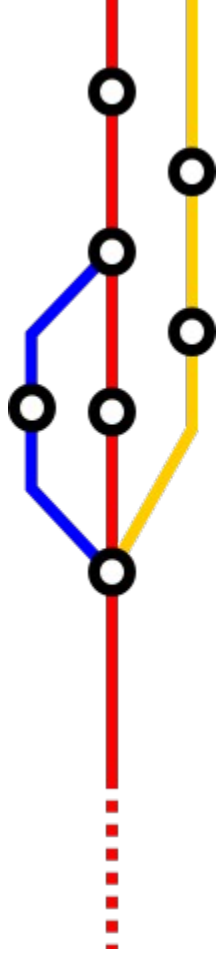
L'artillerie lourde pour avoir un bel affichage de logs :

```
$ git config alias.hist \
log --pretty=format:'%h %ad | %s%d [%an]' --graph --
date=short
```


CC BY SA

Bonus

Petit outil de visualisation



Gource :

https://gource.io/

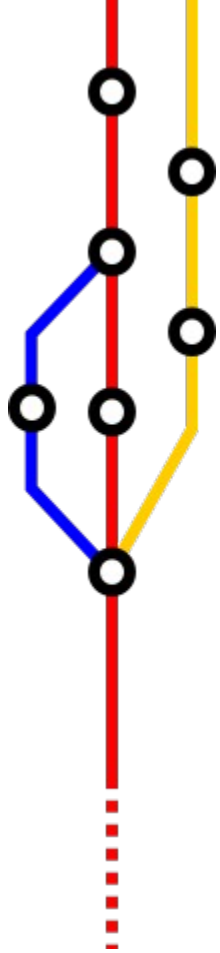
Petit soft rigolo à lancer à la fin d'un projet Git !

Exemple :

https://www.youtube.com/watch?v=rWxFDh0NVKw

CC BY SA

Liens et références



- **Tutoriel Git imagé (EN) :**

  https://rogerdudler.github.io/git-guide/

- **GitHowTo :**

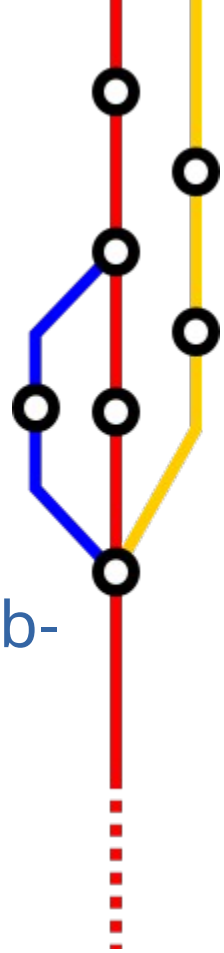
  https://githowto.com/

- **Référence de la doc Git (EN) :**

  https://git-scm.com/docs/

CC BY SA

Liens et références



- **La GitHub cheat sheet (EN) :**

https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf

- **Un modèle de 'workflow' : Gitflow**

https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

- **Le man de Git et les man sous-jacent**

CC BY SA

Merci de votre attention !

CC BY SA

Club Informatique