

Programmation WEB

git pour les novices

Pierre Sudron (et Denis Monnerat)

Updated: 2017/12/09

IUT de Fontainebleau

git est un système de gestion de sources

- gérer l'évolution d'un code
- partager efficacement son travail en équipe
- garder un historique de l'évolution d'un projet
- travailler en parallèle



Autres systèmes existants

- Subversion (svn)
- Mercurial (Hg)
- Bazaar (bzt)
- autres systèmes propriétaires

Plus particulièrement sur git

- créé pour le noyau Linux
- développé par Junio Hamano
- distribué et très flexible
- adoption rapide et massive depuis 2005

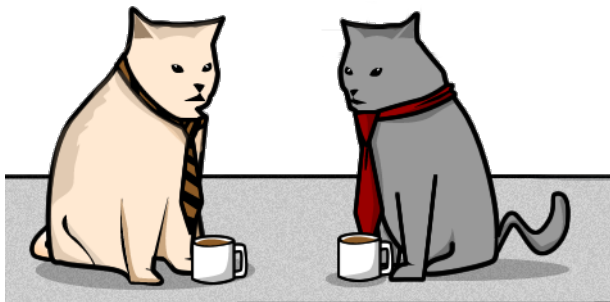


<https://git-scm.com/>

Qu'est-ce que git peut m'apporter ?

- suivi précis de l'avancement d'un projet
- souplesse dans l'évolution
- facilité de mise en commun

Intégration douloureuse : un projet qui fait plouf

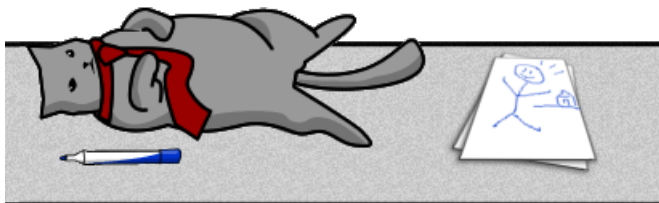


Bob et Bob travaillent sur un projet
et se répartissent les tâches

À deux jours du livrable...

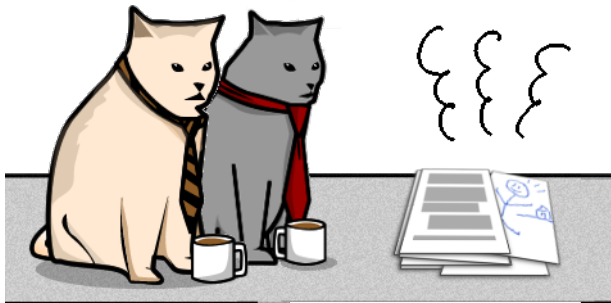


Bob a bien avancé sur sa partie



Bob pas tellement
mais il a fait quelque chose au moins

C'est l'heure de mettre en commun !



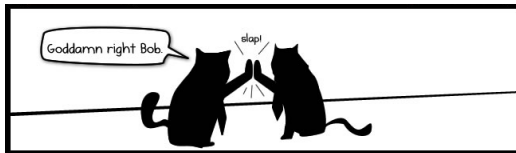
Et quelle surprise, ça marche pas !



Et à 24 heures du rendu, c'est un peu cuit...

L'intégration progressive

- git incite à mettre en commun très régulièrement
- en cas d'ennui, il est possible de revenir à la dernière version fonctionnelle
- de là, il est facile de voir les modifications ultérieures et isoler le code en cause



De quoi ai-je besoin ?

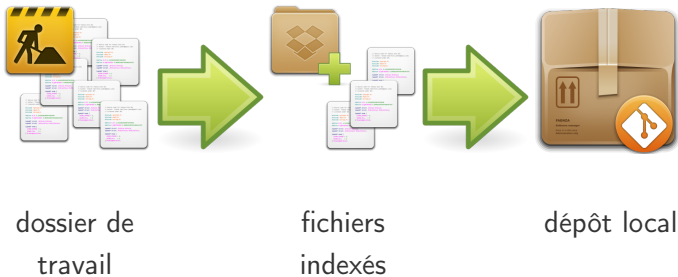
Ouvrir un terminal

(mais sous Linux, hein...)

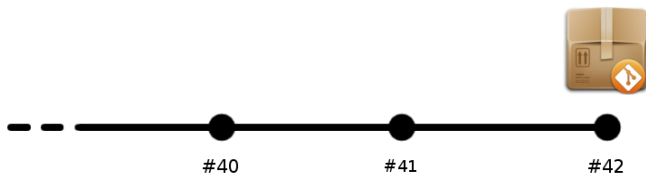


Principes de fonctionnement

En **local**, on travaille avec 3 éléments :



Succession de **commits**



Un commit est un jeu de modifications, c'est l'*atome* de git.

Synchronisation entre le dépôt local et un dépôt distant



Partie 1 : C'est un voyage qu'il faut commencer seul

versionnage d'un simple fichier texte

```
// Source code for Faezra Icon Set
// Author: Tsiheun (matthieu.james@gmail.com)
// Licenced under GPL

#include <gtk/gtk.h>
#include <math.h>
#include <string.h>

#define M_PI 3.14159265358979323846
#define M_SQRT3OVER2 0.8660254037844386467637

typedef struct _RcStyle RcStyle;
typedef struct _RcStyleClass RcStyleClass;

typedef enum {
    CAIRO_STROKE = 1,
    CAIRO_FILL = 2
} DrawingOperation;
```


Configuration de git : qui suis-je ?

git retrace l'évolution du projet ainsi :

qui a fait **quoi**, et **quand**

Pour définir **qui** est l'utilisateur :

```
git config --global user.name "Votre_nom"  
git config --global user.email "root@atilla.org"
```

Pour activer la couleur dans le terminal :

```
git config --global color.ui auto
```

Préparer le projet

Se placer dans le dossier de travail (le créer si besoin) :

```
mkdir ~/formation_git  
cd ~/formation_git/
```

Initialiser git dans ce répertoire :

```
git init
```

Comment ça se présente



répertoire de travail : vide



fichiers indexés : vide



dépôt local : initialisé et vide

Créer un fichier

Sans changer de dossier, créer un fichier **README** avec le contenu suivant :

```
Formation git
```

Enregistrer le fichier

Comment ça se présente



répertoire de travail : README



fichiers indexés : vide



dépôt local : initialisé et vide

Constatez les changements avec git status

```
git status
```

```
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       README
nothing added to commit but untracked files present (use "git add" to track)
```

README n'est pas indexé, on nous propose d'utiliser git add

Indexer un fichier avec git add

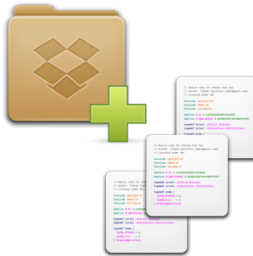
```
git add README
```

```
// Source code for Faucus Icon Set
// Author: Tihuan [tiahzou.jame@gmail.com]
// Licensed under GPL

#include <git/git.h>
#include <math.h>
#include <string.h>

void* fnc_0x1315905058097932046
void* fnc_0x1315905058097932046
typedef struct _obj_t obj_t;
typedef struct _obj_t obj_t;

typedef enum {
    CASE_STROKE = 1,
    CASE_FILL = 2
} DrawOperation;
```



Comment ça se présente



répertoire de travail : README



fichiers indexés : README (nouveau fichier)



dépôt local : initialisé et vide

Constater les changements avec git status

```
git status
```

```
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file:   README
#
```

README est dans l'index en tant que nouveau fichier

Notre premier git commit

On va mettre à jour le dépôt local et laisser une trace de la création du fichier README

```
git commit -m "Ajout du fichier README"
```

Tout commit est décrit par un **message**, il doit être :

- précis**
- concis**

Comment ça se présente



répertoire de travail : README



fichiers indexés : vide



dépôt local : commit #1

(vérifier avec git status)

Historique des commits



Poursuivons le travail...

Ajouter une ligne à la fin du fichier **README** :

```
Formation git  
IUT de Fontainebleau
```

Enregistrer le fichier

Comment ça se présente



répertoire de travail : README (modifié)



fichiers indexés : vide



dépôt local : commit #1

Constater les changements avec git status

```
git status
```

Les fichiers modifiés sont détectés ; on propose de les indexer avec git add

```
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   README
#
no changes added to commit (use "git add" and/or "git commit -a")
```

Indexation du fichier modifié

```
git add README
```



répertoire de travail : README (modifié)



fichiers indexés : README (modifié)



dépôt local : commit #1

(vérifier avec git status)

Validation des changements

```
git commit -m "Modification du fichier README"
```



répertoire de travail : README (modifié)



fichiers indexés : vide



dépôt local : commit #1, commit #2

(vérifier avec git status)

Historique des commits



Historique des commits

```
git log
```

```
commit 82aad05924900b273d50f3b55e7d905896931e8d
```

```
Author: Pierre Sudron <sudronpier@eisti.eu>
```

```
Date: Thu Aug 23 20:16:30 2012 +0200
```

```
Modification du fichier README
```

```
commit b1098965de8d0948104ceb657be01fb9a381860a
```

```
Author: Pierre Sudron <sudronpier@eisti.eu>
```

```
Date: Thu Aug 23 18:36:22 2012 +0200
```

```
Ajout du fichier README
```

Export vers un dépôt git en ligne

Nous utiliserons **Gogs** (Go Git Service) hébergé sur un serveur de l'iut :

```
https://dwarves.iut-fbleau.fr/git
```

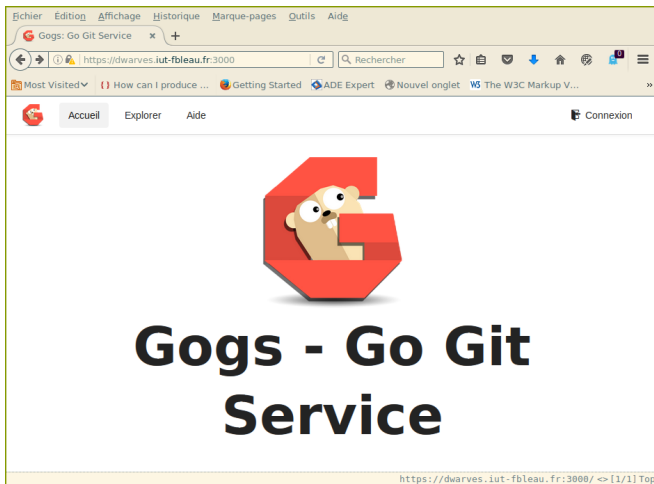


Il existe de nombreux sites permettant d'héberger vos projets git, dont :

- GitLab
- GitHub

Se connecter et ajuster son profil sur Gogs

Le service s'appuie sur le serveur ldap du département.



Se connecter et ajuster son profil sur Gogs

Utilisez vos identifiants du réseau interne de l'iut.

Fichier Edition Affichage Historique Marque-pages Outils Aide

Connexion - Gogs: G... x +

https://dwarves.iut-fbleau.fr:3000/user/login?redirect

Rechercher

Most Visited How can I produce ... Getting Started ADE Expert Nouvel onglet The W3C Markup V...

Accueil Explorer Aide Connexion

Connexion

Nom d'utilisateur ou e-mail * monnerat

Mot de passe *

Se souvenir de moi

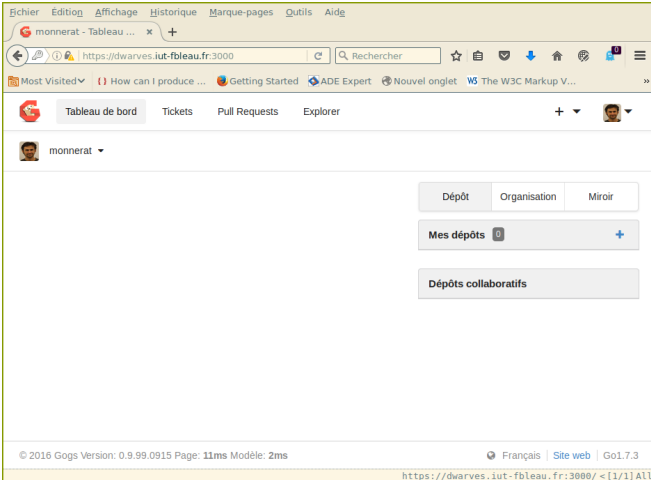
Connexion [Mot de passe oublié ?](#)

© 2016 Gogs Version: 0.9.99.0915 Page: 1ms Modèle: 1ms Français | [Site web](#) | Go1.7.3

INSERT https://dwarves.iut-fbleau.fr:3000/user/login?redirect_to=<[1/1]All

Se connecter et ajuster son profil sur Gogs

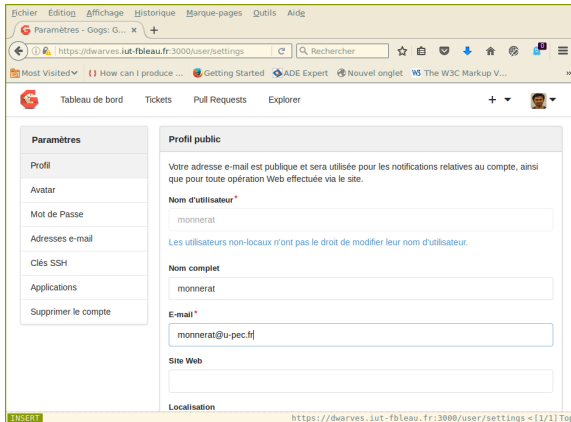
Une fois connecté, vous accédez à votre tableau de bord.



The screenshot shows a web browser window displaying the Gogs dashboard. The browser's address bar shows the URL `https://dwarves.iut-fbleau.fr:3000`. The page title is "monnerat - Tableau de bord". The dashboard includes a navigation menu with "Tableau de bord" (selected), "Tickets", "Pull Requests", and "Explorer". Below the navigation, the user's profile "monnerat" is visible. The main content area features three buttons: "Dépôt", "Organisation", and "Miroir". Below these is a "Mes dépôts" section showing 0 repositories and a "+" button to add more. A "Dépôts collaboratifs" section is also present. The footer contains the text "© 2016 Gogs Version: 0.9.99.0915 Page: 11ms Modèle: 2ms" and "Français | Site web | Go1.7.3". The browser's status bar at the bottom shows the full URL: `https://dwarves.iut-fbleau.fr:3000/<[1/1]All`.

Se connecter et ajuster son profil sur Gogs

Vous pouvez modifier les informations correspondants à votre profil.



The screenshot shows a web browser window with the URL `https://dwarves.iut-fbleau.fr:3000/user/settings`. The page title is "Paramètres - Gogs: G...". The browser's address bar contains the URL and a search box labeled "Rechercher". The page has a navigation bar with "Tableau de bord", "Tickets", "Pull Requests", and "Explorer". The main content area is divided into two columns. The left column, titled "Paramètres", contains a list of settings: "Profil", "Avatar", "Mot de Passe", "Adresses e-mail", "Clés SSH", "Applications", and "Supprimer le compte". The right column, titled "Profil public", contains the following information and form fields:

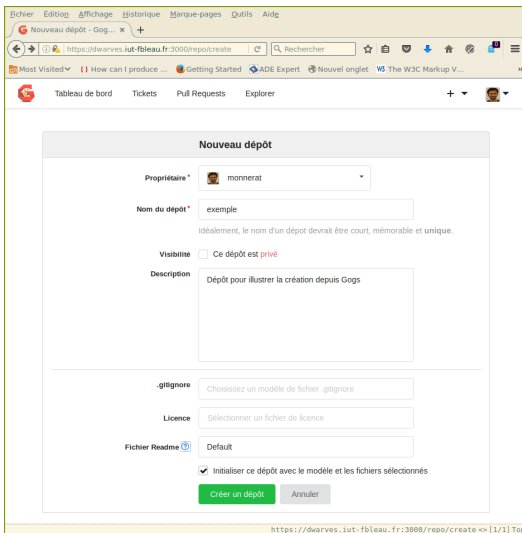
- A note: "Votre adresse e-mail est publique et sera utilisée pour les notifications relatives au compte, ainsi que pour toute opération Web effectuée via le site."
- A field for "Nom d'utilisateur*" containing the text "monnerat". Below it, a blue link reads "Les utilisateurs non-locaux n'ont pas le droit de modifier leur nom d'utilisateur."
- A field for "Nom complet" containing the text "monnerat".
- A field for "E-mail*" containing the text "monnerat@u-pec.fr".
- A field for "Site Web".
- A field for "Localisation".

The browser's status bar at the bottom shows "INSERT" on the left and the URL `https://dwarves.iut-fbleau.fr:3000/user/settings <[1/1] Top` on the right.

Renseignez correctement l'adresse mail.

Créer un projet sur gogs

Un projet peut être public, ou privé.



The screenshot shows the 'Nouveau dépôt' (New Repository) form in a web browser. The browser's address bar shows the URL 'https://dwarves.lut-fbleau.fr:3000/repo/create'. The form is titled 'Nouveau dépôt' and contains the following fields and options:

- Propriétaire ***: A dropdown menu showing the user 'monnerat'.
- Nom du dépôt ***: A text input field containing 'exemple'. Below this field is a note: 'Idéalement, le nom d'un dépôt devrait être court, mémorable et unique.'
- Visible**: A checkbox labeled 'Ce dépôt est privé' which is currently unchecked.
- Description**: A text area containing the text 'Dépôt pour illustrer la création depuis Gogs'.
- .gitignore**: A text input field with the placeholder 'Choisissez un modèle de fichier .gitignore'.
- Licence**: A text input field with the placeholder 'Sélectionner un fichier de licence'.
- Fichier README**: A text input field with the value 'Default'.
- Initialiser ce dépôt avec le modèle et les fichiers sélectionnés
- Créer un dépôt**: A green button.
- Annuler**: A grey button.

The browser's footer shows the URL 'https://dwarves.lut-fbleau.fr:3000/repo/create <> [1/1] Top'.

Créer un projet sur gogs

The screenshot shows a web browser displaying a Gogs repository page for a user named 'monnerat'. The page title is 'monnerat / exemple'. The repository has 1 follower, 0 votes, and 0 forks. The main navigation includes 'Code', 'Tickets (0)', 'Pull Requests (0)', 'Commits (1)', 'Publications (0)', 'Wiki', and 'Paramètres'. The main content area shows a commit by 'monnerat' (03a5069fd2) titled 'Initial commit' made 'il y a 1 seconde'. The commit includes a file named 'README.md' (03a5069fd2) titled 'Initial commit'. The content of the README.md file is displayed below, showing the title 'exemple' and the text 'Dépôt pour illustrer la création depuis Gogs'. The URL at the bottom of the page is 'https://dwarves.iut-fbleau.fr:3000/monnerat/exemple <[1/1] All'.

Fichier Edition Affichage Historique Marque-pages Outils Aide

monnerat/exemple: ... x +

https://dwarves.iut-fbleau.fr:3000/monnerat/exem Recherche

Most Visited How can I produce ... Getting Started ADE Expert Nouvel onglet WS The W3C Markup V...

Tableau de bord Tickets Pull Requests Explorer +

monnerat / exemple Ne plus suivre 1 Voter 0 Fork 0

Code Tickets 0 Pull Requests 0 Commits 1 Publications 0 Wiki Paramètres

Dépôt pour illustrer la création depuis Gogs

Branche: master exemple Nouveau fichier Télécharger un fichier HTTPS SSH https://dwarves.iut-fbleau.fr:3000/monnerat/exemple

monnerat 03a5069fd2 Initial commit il y a 1 seconde

README.md 03a5069fd2 Initial commit il y a 1 seconde

README.md

exemple

Dépôt pour illustrer la création depuis Gogs

https://dwarves.iut-fbleau.fr:3000/monnerat/exemple <[1/1] All

Créer une clé SSH

Ouvrez un nouveau terminal :

```
ssh-keygen -t rsa -C "email@mondomaine.com"
```

La clé SSH permet :

- de vous identifier formellement
- de crypter le transfert de code entre vous et le serveur



Indiquer la clé publique à gogs

Le RSA est un cryptage asymétrique, et crée un jeu de deux clés

- une clé privée
- une **clé publique**

Pour afficher la clé publique :

```
cat .ssh/id_rsa.pub
```

- ajouter la clé publique au trousseau gogs

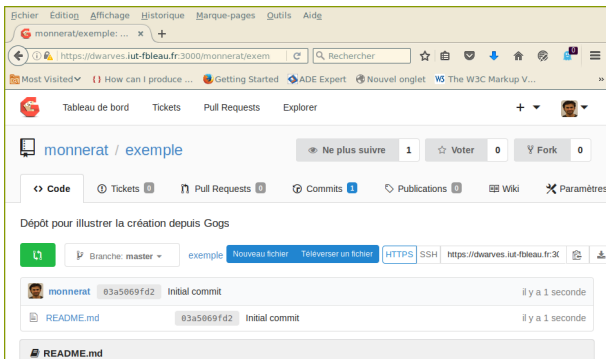
(vous pouvez fermer le second terminal et revenir au précédent)

Ajouter le dépôt dans votre projet

```
git remote add origin [adresse du depot]
```

origin : nom donné au serveur distant par convention

Vous trouverez l'adresse (ssh ou https) de votre dépôt sur sa page d'accueil.



The screenshot shows a web browser displaying the GitHub repository page for 'monnerat / exemple'. The browser's address bar shows the URL 'https://dwarves.iut-fbleau.fr:3000/monnerat/exem'. The repository page includes navigation tabs for 'Code', 'Tickets', 'Pull Requests', 'Commits', 'Publications', 'Wiki', and 'Paramètres'. Below the repository name, there are buttons for 'Ne plus suivre', 'Voter', and 'Fork'. The main content area displays the repository URL 'https://dwarves.iut-fbleau.fr:3000/monnerat/exem' and a list of commits. The first commit is 'Initial commit' by 'monnerat' with hash '03a5069fd2', and the second is 'README.md' with the same hash. Both commits are dated 'il y a 1 seconde'.

Mettre le projet en ligne !

```
git push -u origin master
```

- **origin** : nom donné au serveur distant par convention
- **master** : nom donné au dépôt local par défaut

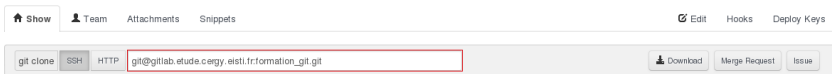


master

origin

Cloner un dépôt existant

- récupérer l'adresse du dépôt



- cloner le dépôt (cette opération crée le répertoire du projet)

```
git clone [adresse]
```

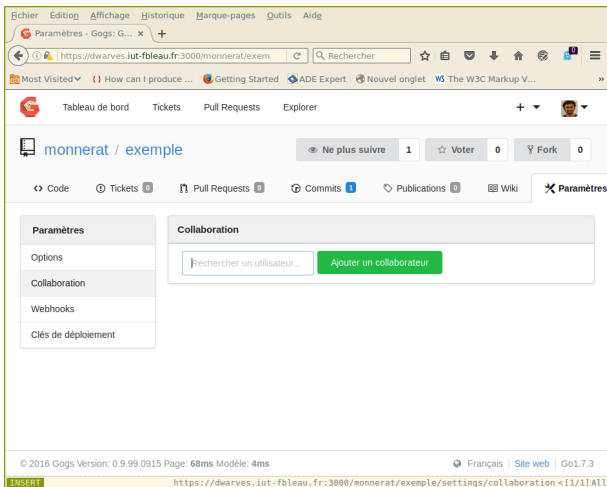
Partie 2 : Travailler à plusieurs

découverte des fonctionnalités de partage



Accès aux projets

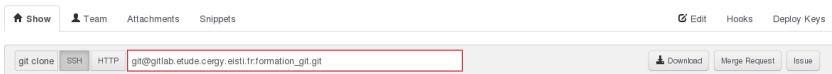
Sous gogs, le projet est public ou privé. Le propriétaire peut rajouter des collaborateurs :



The screenshot shows a web browser window displaying the Gogs interface. The address bar shows the URL `https://dwarves.iut-fbleau.fr:3000/monnerat/exem`. The page title is `monnerat / exemple`. The navigation bar includes `Tableau de bord`, `Tickets`, `Pull Requests`, and `Explorer`. The main content area shows the `Collaboration` settings for the repository. A search input field labeled `Rechercher un utilisateur...` is present, along with a green `Ajouter un collaborateur` button. The footer displays `© 2016 Gogs Version: 0.9.99.0915 Page: 68ms Modèle: 4ms` and `Français | Site web | Go1.7.3`. The browser's status bar at the bottom shows `INSERTE` and the full URL `https://dwarves.iut-fbleau.fr:3000/monnerat/exemple/settings/collaboration`.

Cloner un dépôt existant

- récupérer l'adresse du dépôt



- cloner le dépôt (cette opération crée le répertoire du projet)

```
git clone [adresse]
```

Cloner un dépôt existant

Le projet **Participants formation** contient un seul fichier dans lequel chacun va ajouter son nom.

- assurez-vous que vous êtes inscrit au projet comme développeur
- clonez le dépôt sur votre machine
- tout le monde doit être à la même version

Allons y...

- ajoutez votre nom à la fin du fichier
- faites un commit
- essayer de push...

```
To git@gitlab.etude.cergy.eisti.fr:formation_git.git
! [rejected]      master -> master (non-fast-forward)
error: failed to push some refs to 'git@gitlab.etude.cergy.eisti.fr:formation_git.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Merge the remote changes (e.g. 'git pull')
hint: before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

oui, ça veut dire
PA - TA - TRAS !

Qu'est-ce que j'ai fait pour mériter ça ?

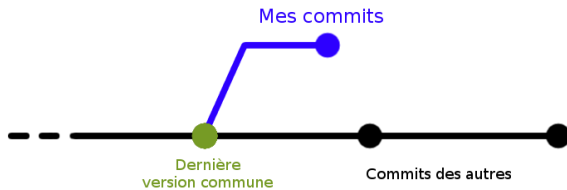
Si il y a un mot à retenir dans le message d'erreur :

```
! [rejected]          master -> master (non-fast-forward)
```



non-fast-forward signifie que vous écraseriez les commits d'autres si git vous laissait push.

Qu'est-ce qui s'est passé ?



Un fichier a été modifié et mis en ligne depuis votre dernier pull.

Comment s'en sortir ?

On souhaiterait appliquer nos commits à la suite de ceux des autres.



C'est une opération de **rebase**.

Chacun son tour

- récupérer les derniers commits en ligne et rebase nos commits à partir de là

```
git pull --rebase
```



- cette opération peut soulever un **conflit** bloquant car plusieurs personnes ont modifié le même fichier

Les conflits

Un **conflit** a lieu quand on cherche à mettre en commun deux fichiers dont les mêmes sections ont été modifiées par deux personnes.

Bonjour les gens.		Bonjour les gens.
Il fait beau aujourd'hui.	→ ←	Il fait moche aujourd'hui.
J'aime le sport.	→ ←	Je mange au McDo tous les jours.
Je mange équilibré.		

Résoudre un conflit bloquant un rebase

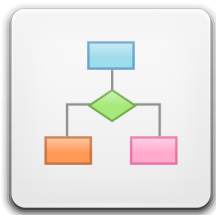
Le message d'erreur du "pull --rebase nous indique les deux étapes à suivre :

- corriger les conflits dans les fichiers continus
- achever et valider le rebase avec

```
git rebase --continue
```

Partie 3 : Comment ça va vieille branche ?

développer en parallèle



C'est l'histoire d'un prof...

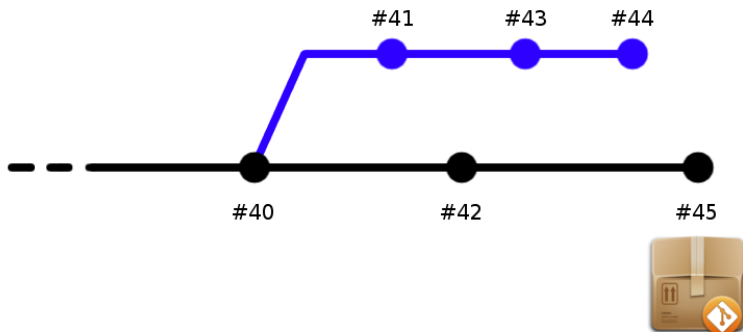
Jean-Paul a fini de rédiger le poly du cours de cette année, en $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ bien entendu !

Il souhaite continuer à rajouter des parties pour l'année suivante tout en gardant une version de cette année afin de corriger les coquilles que ses élèves lui signalent.

Mais comme Jean Paul n'est pas très organisé, nous allons l'aider à mieux gérer son texte de cours avec **git** et les **branches** !

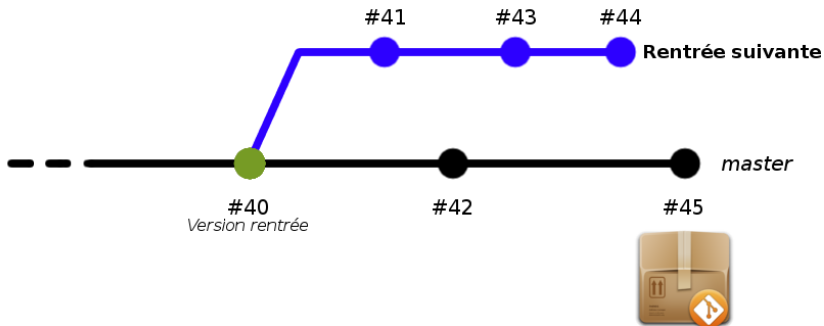
Le principe des branches

Fourche dans le processus de développement du projet. Une branche démarre à partir d'un commit donné.



Le principe des branches

Nous allons créer une branche dédiée aux corrections des fautes d'orthographe sur une branche partant de la version distribuée du poly.



Préparation du projet

- créer un dossier `formation_git_branches`
- déplacer le fichier `cours.tex` fourni dans ce dossier

```
cd formation_git_branches
```

```
git init
```

```
git add cours.tex
```

```
git commit -m "Version_1_rentree"
```

Créer une branche

La branche est créée à partir du dernier commit local (HEAD).

```
git branch rentrée_suivante
```

Pour lister les branches existantes :

```
git branch
```

```
* master  
  rentrée_suivante
```


Sauter de branche en branche

```
git checkout rentrée_suivante
```

La commande `git branch` donne le résultat suivant :

```
master  
* rentrée_suivante
```

Ajouter un nouveau paragraphe pour la rentrée prochaine

Jean-Paul, studieux comme toujours, ne tarde pas à rajouter du contenu pour l'année prochaine.

Éditer le fichier `cours.tex` en ajoutant une ligne à la fin.
Sauvegarder, fermer l'éditeur de texte et commiter.

```
git add cours.tex  
git commit -m "Nouveau paragraphe"
```

Revenir sur la branche principale pour une correction

Mince! On a signalé la présence de fautes d'orthographe dans le poly de cette année.

Retourner sur la branche principale

```
git checkout master
```

Ouvrir le fichier `cours.tex` avec un éditeur. Vérifier qu'il s'agit bien de la version de cette rentrée.

Corriger les fautes, enregistrer, commiter.

Visualiser les commits selon leur branche

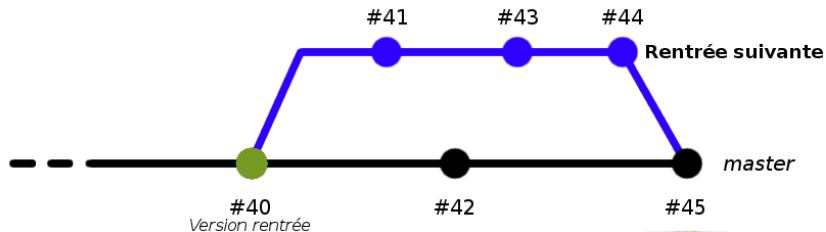
```
git log --oneline --graph --all
```

```
* 7fe9bbb Correction de fautes de frappe  
| * 45e6ac9 Nouveau paragraphe  
|/  
* 194a5b4 Version rentrée
```

le log se lit de bas en haut

Fusionner deux branches

La rentrée suivante est arrivée (ça passe vite), et Jean-Paul voudrait intégrer ses nouvelles parties tout en conservant les corrections faites en cours d'année.



Fusionner deux branches

Se placer dans la branche qui va "intégrer" les commits de l'autre

```
git checkout master
```

Réaliser la fusion

```
git merge rentree_suivante
```

... et là : **Merge conflict !**

```
Auto-merging cours.tex
CONFLICT (content): Merge conflict in cours.tex
Automatic merge failed; fix conflicts and then commit the result.
```

Certains paragraphes ont un contenu différent, il va falloir gérer ça à la main.

Gérer un merge conflict

Ouvrir le fichier `cours.tex`. git a modifié son contenu pour mettre en évidence le conflit.

```
<<<<<< HEAD
    ancien contenu
=====
    nouveau contenu
>>>>>> rentree_suivante
```

git status permet de voir sur quels fichiers il existe des conflits :

```
# On branch master
# Unmerged paths:
#   (use "git add/rm <file>..." as appropriate to mark resolution)
#
#       both modified:   cours.tex
#
```

Gérer un merge conflict

La démarche à suivre pour résoudre le conflit :

- corriger les conflits dans chaque fichier concerné
- vérifier que tous les conflits sont corrigés avec **git status**
- indexer (**git add**) tous les fichiers modifiés dans la manipulation
- commiter (**git commit**), on appelle ça un *'merge commit'*

Corriger des conflits dans un fichier plus facilement

Il est assez compliqué de s'en sortir avec ça :

```
<<<<<< HEAD
          ancien contenu
=====
          nouveau contenu
>>>>>> rentree_suivante
```

Finaliser la résolution de conflits

```
git status
git add cours.tex
git commit -m "Merge dans master de rentree_suivan"
```

Mettre en ligne votre branche

```
git push origin [ma_branche]
```



ma_branche

origin

Terminer une branche

```
git branch -d rentree_suivante
```

Il est possible de terminer seulement une branche qui a été mergée et n'a pas été modifiée depuis. Pour forcer la suppression d'une branche :

```
git branch -D rentree_suivante
```

Partie 3 : remonter le temps

C'est moi Doc ! Je suis de retour du futur...



Accès à un commit

- **HEAD** : dernier commit de la branche courante
- **nom_branche/HEAD** : dernier commit de la branche nom_branche
- donner le nom d'une branche revient à pointer vers le dernier commit de cette branche

Il est possible de se "déplacer" relativement à un commit

- **HEAD^** : un commit avant HEAD
- **HEAD^^** : deux commits avant HEAD
- **HEAD~42** : quarante deux commits avant HEAD

Accès à un commit

- trouver l'identifiant court d'un commit

```
git log --oneline
```

- `master@{20/09/2012}` : prendre un commit à une date donnée

Voyager dans le temps

- mettre un fichier tel qu'il était à un commit donné

```
git checkout [nom_commit] [fichier]
```

Cette manipulation modifie directement le fichier dans votre répertoire de travail.

- remettre un fichier "en l'état"

```
git checkout HEAD [fichier]
```


Annexe 1 : Quelques conseils

tout va bien se passer...



Commitez bien, commitez souvent

- segmentez les tâches, faites un commit dès que possible
- ne commitez pas sciemment un code qui ne marche pas

Les branches c'est bon, mangez-en !



- séparez les tâches indépendantes
- n'hésitez pas à faire des expérimentations dans une branche dédiée
- gardez une branche "stable" à tout moment de votre développement

Jouez collectif

- organisez-vous et concertez-vous avec vos partenaires
- donnez des titres compréhensibles à vos commits
- suivez les avancées des autres, donnez votre avis
- construisez votre cycle de développement intelligemment : en fonction de la taille et la nature de votre équipe, vos deadlines, *etc.*

Annexe 2 : Guide de survie

Comme Rambo, mais en mieux.



Au début de la journée de travail

- je commence toujours par récupérer le travail des autres

```
git pull
```

- je vérifie sur quelle branche je me trouve

```
git branch  
git checkout
```

Quand je code

- j'ajoute mes fichiers modifiés à l'index

```
git add
```

- je vérifie mes modifications suivies

```
git status
```

- je valide mes changements dans un commit

```
git commit
```

Partager mon travail

- je publie mon travail au moins en fin de journée

```
git push
```

- en cas de fast-forward, j'applique mes modifications après celles des autres

```
git pull --rebase
```

- en cas de conflit, je fait les modications à la main

```
git mergetool  
git rebase --continue
```


Au secours, rien ne va plus !

- faire attention à ce qu'on fait pour éviter les ennuis inutiles
- DON'T PANIC!
- prendre le temps de lire les messages d'erreur de git (ils donnent souvent la solution)
- il existe de très nombreuses ressources sur le net