Génie logiciel : introduction à GIT

Karën Fort

karen.fort@univ-lorraine.fr / https://members.loria.fr/KFort

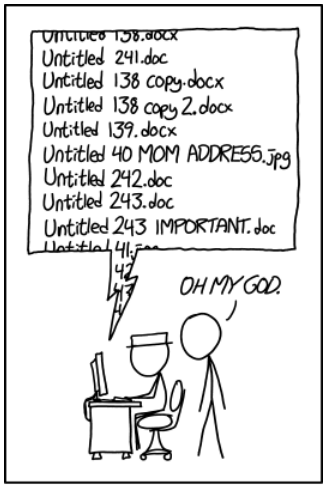
Institut des sciences du Digital Management & Cognition
COMPOSANTE DE L'UNIVERSITÉ DE LORRAINE

UNIVERSITÉ DE LORRAINE

Quelques sources d'inspiration

- https://www.w3schools.com/git/
- https://git-scm.com/book/fr/v2
- http://ndpsoftware.com/git-cheatsheet.html
- https://devopedia.org/git
- https://c.sambuz.com/899744/5fa8c71b3ac0f.pdf
- une excellente vidéo : https://www.youtube.com/watch?v=8JJ101D3knE

Un système de gestion de version pour un·e, pourquoi faire ?



- en local :
  - versionnage
  - documentation des changements
  - possibilité de revenir en arrière facilement
- en distanciel (par ex. sur GitHub) :
  - sauvegarde

Un système de gestion de version à plusieurs, évidemment

- versionnage
- documentation des changements
- possibilité de revenir en arrière facilement
- sauvegarde du travail

\+ possibilité de travailler en parallèle sur un projet
  - savoir qui a fait quoi
  - fusionner le travail de plusieurs personne sur un ou plusieurs fichiers

GIT vs GitHub vs GitLab

	Logiciel de gestion de version	Libre (GPLv2)
	Plateforme (serveur hôte + outils)	Microsoft
	Plateforme (serveur hôte + outils+)	Libre (coeur)

Exercice : installer Git

https://git-scm.com/downloads

NB : si vous êtes sur Windows, utilisez `Git bash` (fourni) pour tous nos exercices

Exercice : configurer Git

Il faut donner à Git certaines infos vous concernant :

- `git config --global user.name "vos prénom et nom"`
- `git config --global user.email monmail@monserveur.fr`
- `git config --global --global core.editor "votreEditeur --wait"`
- `git config --global --global core.autocrlf auto (ou input si pas Windows)`

Exercice : notre espace de travail

`https://gitlab.univ-lorraine.fr/` :

- Connectez-vous avec vos identifiants de l'université
- Demandez à accéder au projet m1_miage_2024-25 (`Search or go to...`)
- Explorez :
  - Votre profil
  - Le projet

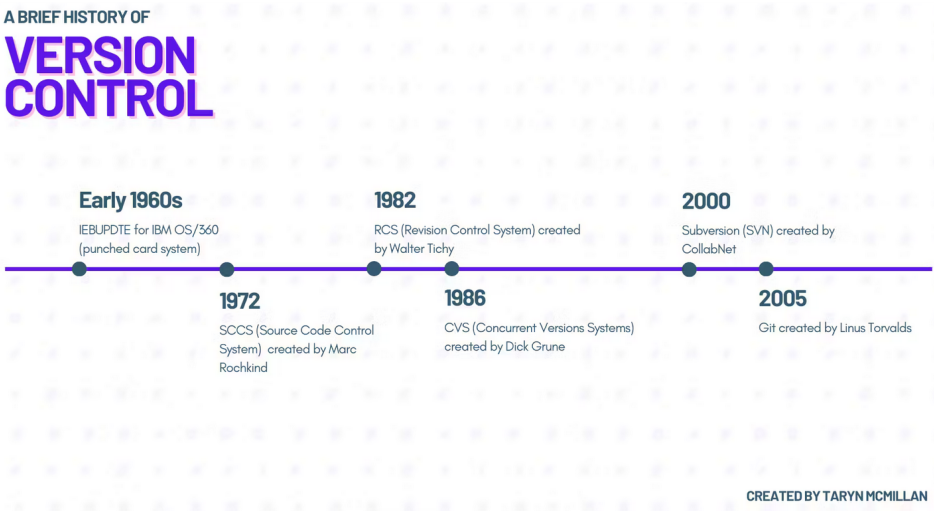
Exercice : cloner le dépôt Git

En ligne de commande :

- positionnez-vous à l'endroit voulu sur votre disque
- créez un répertoire (par ex : CoursM1Git)
- clonez le dépôt Git que j'ai créé :
  git clone https://gitlab.univ-lorraine.fr/fort5/m1_miage_2024-25.git
- explorez le dépôt sur votre disque dur

Git Clone

```
karen@Delbo:~/Documents/Cours/CoursPR_Nancy/M1_GenieLogiciel/Cours_git$ git clone
 https://gitlab.univ-lorraine.fr/fort5/m1_miage_2024-25.git
Clonage dans 'm1_miage_2024-25'...
Username for 'https://gitlab.univ-lorraine.fr': fort5
Password for 'https://fort5@gitlab.univ-lorraine.fr':
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 (from 0)
Réception d'objets: 100% (3/3), fait.
karen@Delbo:~/Documents/Cours/CoursPR_Nancy/M1_GenieLogiciel/Cours_git$ cd m1_mia
ge_2024-25/
karen@Delbo:~/Documents/Cours/CoursPR_Nancy/M1_GenieLogiciel/Cours_git/m1_miage_2
024-25$ ls
README.md
```



https://blog.tarynmcmillan.com/a-history-of-version-control

Trois générations de systèmes

	Réseau	Opérations	Concurrence	Exemples
1972-1982	Aucun	1 fichier à la fois	Verrous (locks)	SCCS, RCS
1986-2000	Centralisé	Multi fichier	Merge avant commit	CVS, Subversion
2005-	Distribué	Instantanés	Commit avant merge	Git, Mercurial

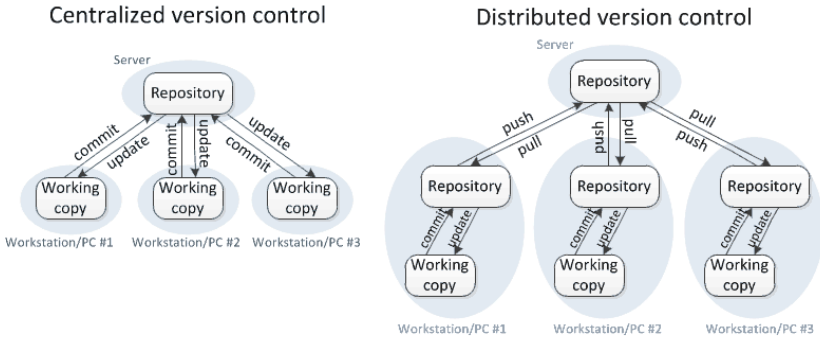
Adapté de https://ericsink.com/vcbe/html/history_of_version_control.html

Trois générations de systèmes

	Réseau	Opérations	Concurrence	Exemples
1972-1982	Aucun	1 fichier à la fois	Verrous (locks)	SCCS, RCS
1986-2000	Centralisé	Multi fichier	Merge avant commit	CVS, Subversion
2005-	Distribué	Instantanés	Commit avant merge	Git, Mercurial

Adapté de https://ericsink.com/vcbe/html/history_of_version_control.html

Centralisé (SVN) vs Distribué (GIT)



https://devopedia.org/git

GIT : conséquences

- chaque développeur a en local un historique des versions (*repository*) en plus de sa version de travail (*working copy*) $\rightarrow$ moins de risque de tout perdre (robustesse)
- presque toutes les opérations sont locales (rapidité)

Trois générations de systèmes

	Réseau	Opérations	Concurrence	Exemples
1972-1982	Aucun	1 fichier à la fois	Verrous (locks)	SCCS, RCS
1986-2000	Centralisé	Multi fichier	Merge avant commit	CVS, Subversion
2005-	Distribué	Instantanés	Commit avant merge	Git, Mercurial

Adapté de https://ericsink.com/vcbe/html/history_of_version_control.html

GIT : conséquences

"Git pense ses données plus comme un instantané d'un mini système de fichiers. À chaque fois que vous validez ou enregistrez l'état du projet dans Git, il prend effectivement un instantané du contenu de votre espace de travail à ce moment et enregistre une référence à cet instantané. Pour être efficace, si les fichiers n'ont pas changé, Git ne stocke pas le fichier à nouveau, juste une référence vers le fichier original qu'il a déjà enregistré."

https://git-scm.com/book/fr/v2/D%C3%A9marrage-rapide-Rudiments-de-Git

Conclusion

Avoir l'historique des versions en local n'est pas forcément une catastrophe en terme de place disque...

Un peu plus loin...

Git gère l'intégrité :

- tout est vérifié par une somme de contrôle (empreinte SHA-1) avant d'être stocké

→ il est impossible de modifier le contenu d'un fichier ou d'un répertoire sans que Git ne s'en aperçoive

https://git-scm.com/book/fr/v2/D%C3%A9marrage-rapide-Rudiments-de-Git

In case of fire

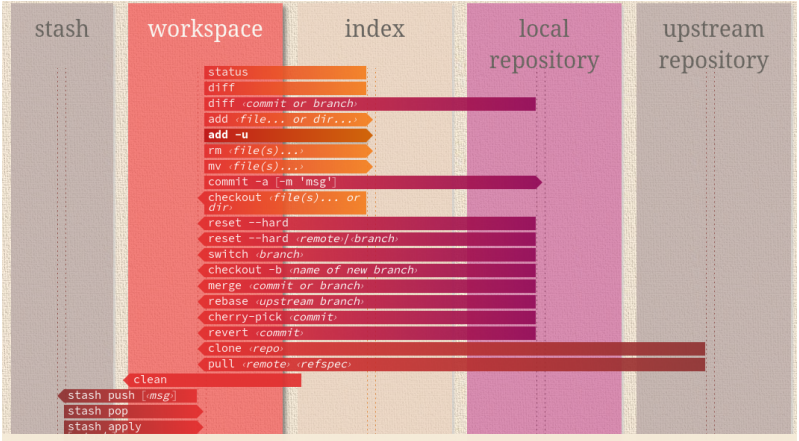
1. git commit
2. git push
3. leave building

Exercice : premier fichier sur Git

En local, dans `m1_miage_2024-25/Etudiants` :

- créez un fichier .txt dont le nom suit le format suivant :
  `VotreNom_VotrePrenom.csv` (évitez les diacritiques et les espaces)
- éditez ce fichier avec un éditeur de texte (de qualité, pas notepad) et ajoutez-y les informations suivantes, séparées par des virgules :
  - Nom
  - Prénom
  - Votre niveau en Git (de 0, débutant, à 10, expert)
- enregistrez (utf-8) et fermez
- vérifiez que le fichier est bien créé : `ls`

L'espace de travail (`workspace`)



workspace

Local checkout of your code. Also called 'working copy', 'working tree' or just 'checkout'. It's any directory on your filesystem that has a repository associated with it (typically indicated by the presence of a sub-directory within it named `.git`). It includes all the files and sub-directories in that directory.

http://ndpsoftware.com/git-cheatsheet.html#loc=workspace;

Git status

```
karen@Delbo:~/Documents/Cours/CoursPR_Nancy/M1_GenieLogiciel/Cours_git/m1_miage_2
024-25$ gedit FortKaren.csv
karen@Delbo:~/Documents/Cours/CoursPR_Nancy/M1_GenieLogiciel/Cours_git/m1_miage_2
024-25$ ls
Etudiants  FortKaren.csv  README.md
karen@Delbo:~/Documents/Cours/CoursPR_Nancy/M1_GenieLogiciel/Cours_git/m1_miage_2
024-25$ git status
Sur la branche main
Votre branche est à jour avec 'origin/main'.

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

aucune modification ajoutée à la validation mais des fichiers non suivis sont pré
sents (utilisez "git add" pour les suivre)
```

Les fichiers peuvent être :

- "suivis" : ils sont dans le dépôt
- "non suivis" : les fichiers sont dans le répertoire de travail, mais pas dans le dépôt

Git status

```
karen@Delbo:~/Documents/Cours/CoursPR_Nancy/M1_GenieLogiciel/Cours_git/m1_miage_2
024-25$ gedit FortKaren.csv
karen@Delbo:~/Documents/Cours/CoursPR_Nancy/M1_GenieLogiciel/Cours_git/m1_miage_2
024-25$ ls
Etudiants  FortKaren.csv  README.md
karen@Delbo:~/Documents/Cours/CoursPR_Nancy/M1_GenieLogiciel/Cours_git/m1_miage_2
024-25$ git status
Sur la branche main
Votre branche est à jour avec 'origin/main'.

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

aucune modification ajoutée à la validation mais des fichiers non suivis sont pré
sents (utilisez "git add" pour les suivre)
```

On fait quoi ensuite ?

Git status

```
karen@Delbo:~/Documents/Cours/CoursPR_Nancy/M1_GenieLogiciel/Cours_git/m1_miage_2
024-25$ gedit FortKaren.csv
karen@Delbo:~/Documents/Cours/CoursPR_Nancy/M1_GenieLogiciel/Cours_git/m1_miage_2
024-25$ ls
Etudiants  FortKaren.csv  README.md
karen@Delbo:~/Documents/Cours/CoursPR_Nancy/M1_GenieLogiciel/Cours_git/m1_miage_2
024-25$ git status
Sur la branche main
Votre branche est à jour avec 'origin/main'.

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

aucune modification ajoutée à la validation mais des fichiers non suivis sont pré
sents (utilisez "git add" pour les suivre)
```

On fait quoi ensuite ?

on lit les instructions à l'écran !

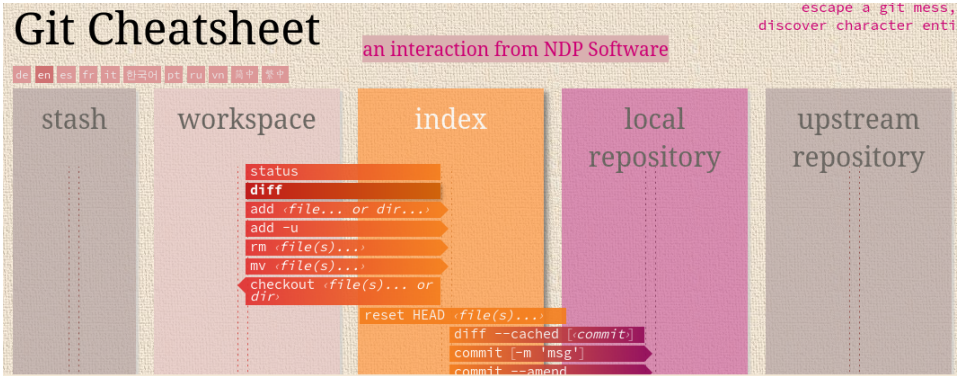
Git add

```
karen@Delbo:~/Documents/Cours/CoursPR_Nancy/M1_GenieLogiciel/Cours_git/m1_miage_2
024-25$ git add FortKaren.csv
karen@Delbo:~/Documents/Cours/CoursPR_Nancy/M1_GenieLogiciel/Cours_git/m1_miage_2
024-25$ git status
Sur la branche main
Votre branche est à jour avec 'origin/main'.

Modifications qui seront validées :
  (utilisez "git restore --staged <fichier>..." pour désindexer)
        nouveau fichier : FortKaren.csv
```

- le fichier a été ajouté dans l'index : *staged*
- attention : "add" peut être trompeur car on peut "ajouter" des suppressions de fichiers dans l'index

L'espace index



http://ndpsoftware.com/git-cheatsheet.html#loc=index;

Git commit

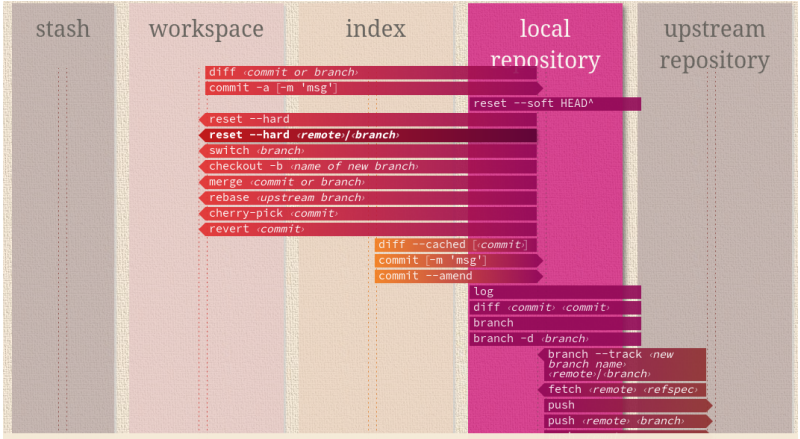
```
karen@Delbo:~/Documents/Cours/CoursPR_Nancy/M1_GenieLogiciel/Cours_git/m1_miage_2
024-25$ git commit -m "ajout du fichier de présentation de Karen Fort"
[main b82bd88] ajout du fichier de présentation de Karen Fort
 1 file changed, 1 insertion(+)
 create mode 100644 FortKaren.csv
karen@Delbo:~/Documents/Cours/CoursPR_Nancy/M1_GenieLogiciel/Cours_git/m1_miage_2
024-25$ git status
Sur la branche main
Votre branche est en avance sur 'origin/main' de 1 commit.
  (utilisez "git push" pour publier vos commits locaux)

rien à valider, la copie de travail est propre
```

- le fichier a été "commité" (ajouté) dans le dépôt local
- -m permet d'ajouter un message :
  - explicite
  - précis
  - complet

Les messages de commit : la tendance

Le dépôt local (`local repository`)



http://ndpsoftware.com/git-cheatsheet.html#loc=local_repo;

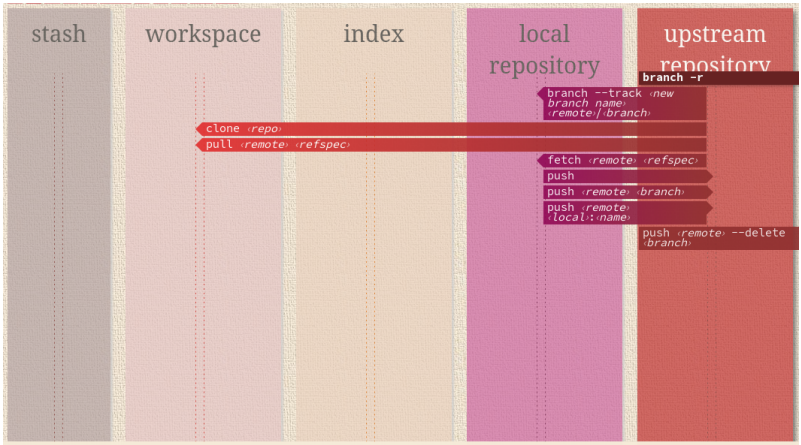
Git push

```
karen@Delbo:~/Documents/Cours/CoursPR_Nancy/M1_GenieLogiciel/Cours_git/m1_miage_2
024-25$ git push
Username for 'https://gitlab.univ-lorraine.fr': fort5
Password for 'https://fort5@gitlab.univ-lorraine.fr':
Énumération des objets: 4, fait.
Décompte des objets: 100% (4/4), fait.
Compression par delta en utilisant jusqu'à 12 fils d'exécution
Compression des objets: 100% (2/2), fait.
Écriture des objets: 100% (3/3), 318 octets | 318.00 Kio/s, fait.
Total 3 (delta 0), réutilisés 0 (delta 0), réutilisés du pack 0
To https://gitlab.univ-lorraine.fr/fort5/m1_miage_2024-25.git
   9107a26..b82bd88  main -> main
karen@Delbo:~/Documents/Cours/CoursPR_Nancy/M1_GenieLogiciel/Cours_git/m1_miage_2
024-25$ git status
Sur la branche main
Votre branche est à jour avec 'origin/main'.

rien à valider, la copie de travail est propre
```

- le fichier a été "pushé" (poussé) sur le serveur (d'où la connexion)

Le dépôt distant (`upstream repository`)



upstream repository

A repository of your code to share and collaborate with other developers. It's hosted on some the Internet or a remote, eg. Github. The default name is `origin`. Typical branches here: `main`, `master`, `shared-feature-x`, `release-y`. Also called 'remote repository', or just 'remote'.

Chacun récupère les ajouts des autres : `git pull`

```
karen@Delbo:~/Documents/Cours/CoursPR_Nancy/M1_GenieLogiciel/Cours_git/m1_miage_2
024-25$ git pull
Username for 'https://gitlab.univ-lorraine.fr': fort5
Password for 'https://fort5@gitlab.univ-lorraine.fr':
Déjà à jour.
```

- on récupère tout ce qui est nouveau sur le serveur (d'où la connexion)
- c'est fini !

Git status version courte

```
karen@Delbo:~/Documents/Cours/CoursPR_Nancy/M1_GenieLogiciel/Cours_git/m1_miage_2
024-25$ gedit FortKaren.csv
karen@Delbo:~/Documents/Cours/CoursPR_Nancy/M1_GenieLogiciel/Cours_git/m1_miage_2
024-25$ git status --short
 M FortKaren.csv
```

- ?? : fichiers non suivis
- A : fichiers ajoutés à l'index
- M : fichers modifiés
- D : fichiers supprimés

Git commit sans add

```
karen@Delbo:~/Documents/Cours/CoursPR_Nancy/M1_GenieLogiciel/Cours_git/m1_miage_2
024-25$ git commit -a -m "Modification mineure (niv en Git)"
[main 55118a9] Modification mineure (niv en Git)
 1 file changed, 1 insertion(+), 1 deletion(-)
karen@Delbo:~/Documents/Cours/CoursPR_Nancy/M1_GenieLogiciel/Cours_git/m1_miage_2
024-25$ git status --short
karen@Delbo:~/Documents/Cours/CoursPR_Nancy/M1_GenieLogiciel/Cours_git/m1_miage_2
024-25$ git status
Sur la branche main
Votre branche est en avance sur 'origin/main' de 1 commit.
  (utilisez "git push" pour publier vos commits locaux)

rien à valider, la copie de travail est propre
```

- `git status --short` ne montre rien
- `git status` montre qu'il manque le `push`

! risque de commiter des changements non voulus

Big brother : `Git log`

```
karen@Delbo:~/Documents/Cours/CoursPR_Nancy/M1_GenieLogiciel/Cours_git/m1_miage_2
024-25$ git log
commit 55118a9cfb86f1f45f6337d099888c2056b1252f (HEAD -> main)
Author: Karen Fort <karen.fort@loria.fr>
Date:   Sat Feb 1 16:06:04 2025 +0100

    Modification mineure (niv en Git)

commit b82bd887cb04c596e8c704d3d15d976cfc6bac4b (origin/main, origin/HEAD)
Author: Karen Fort <karen.fort@loria.fr>
Date:   Sat Feb 1 14:40:17 2025 +0100

    ajout du fichier de présentation de Karen Fort

commit 9107a26e316d77ad42e39e04e7db4f573e477b4d
Author: FORT Karen <karen.fort@univ-lorraine.fr>
Date:   Sat Feb 1 12:42:10 2025 +0000

    Initial commit
```

Big brother's father : `Git blame`

```
karen@Delbo:~/Documents/Semagramme/ra-2024$ git blame 12_02-BIB-READONLY-year-publications.bib
00000000 (Not Committed Yet 2025-02-09 10:11:00 +0100   1) % Encoding: UTF-8
e372b47a (Bruno Guillaume   2025-01-14 16:34:27 +0100   2) @inproceedings{boulanger:hal-04623034,
e372b47a (Bruno Guillaume   2025-01-14 16:34:27 +0100   3)     author = {Boulanger, Hugo and Hiebel,
Aurélie},
e372b47a (Bruno Guillaume   2025-01-14 16:34:27 +0100   4)     hal_type = {COMM},
e372b47a (Bruno Guillaume   2025-01-14 16:34:27 +0100   5)     hal_type_title = {Conference papers},
e372b47a (Bruno Guillaume   2025-01-14 16:34:27 +0100   6)     title = {Génération contrôlée de cas
turées},
e372b47a (Bruno Guillaume   2025-01-14 16:34:27 +0100   7)     halref = {boulanger:hal-04623034},
e372b47a (Bruno Guillaume   2025-01-14 16:34:27 +0100   8)     type_label = {National peer-reviewed
e372b47a (Bruno Guillaume   2025-01-14 16:34:27 +0100   9)     type_number = {050},
e372b47a (Bruno Guillaume   2025-01-14 16:34:27 +0100  10)     presort = {050},
e372b47a (Bruno Guillaume   2025-01-14 16:34:27 +0100  11)     eventtitle = {35èmes Journées d'Étude
ment Automatique des Langues Naturelles (TALN 2024) 26ème Rencontre des Étudiants C:
```

Git init

Commande qui permet de créer son propre dépôt Git (si vous ne clonez pas un dépôt distant) :

https://www.youtube.com/watch?v=A-4WltCTVms

.git

https://www.youtube.com/watch?v=A-4WltCTVms

Exercice : préparation

L'un·e d'entre vous :

- crée un fichier Etudiants.csv qui contient les titres de colonnes informations suivants, séparés par des virgules :
  - Numéro étudiant
  - Nom
  - Prénom
  - Apprenti ?
  - Niveau en Git
- Git :
  - add
  - commit -m "Création du fichier Etudiants"
  - push

Exercice : premier fichier collectif sur Git

Chacun·e d'entre vous doit maintenant remplir ce fichier !

- les différences avec les autres systèmes de gestion de version
- les différences entre Git, GitHub et GitLab
- les commandes Git de base