Git et GitLab

Nicolas Delestre

INSA ROUEN NORMANDIE

Avant propos

- Tous les exemples et illustrations de ce cours sont issus du livre “Pro Git”
  `https://git-scm.com/book/fr/v2`
- Quelques parties de ce cours ont été rédigées à l’aide de ChatGPT

INSA ROUEN NORMANDIE

Constats

- Un ingénieur spécifie, modélise, conçoit et réalise des livrables
  - En ITI ces livrables sont des logiciels, bibliothèques, documents
- Un ingénieur travaille en équipe
  - En ITI chaque membre de l'équipe développe, rédige des parties du livrable
- Un ingénieur gère des équipes, des projets
  - En ITI l'ingénieur affecte des tâches aux membres de l'équipes, sait qui doit faire quoi, sait qui a fait quoi, sait ce qu'il doit faire et dans quel *timing*

Besoin d'un outil efficace...

- pour développer de manière collaborative et traçable $\Rightarrow$ logiciel de gestion de versions
- pour gérer le projet $\Rightarrow$ forge

ROUEN NORMANDIE

Logiciel de gestion de versions 1 / 2

Principes

- **Versionnement des fichiers** : Suivi des modifications dans le temps
- **Concurrence** : Modification concurrente des fichiers par plusieurs utilisateurs
- **Branches** : Versions parallèles du code
- **Fusion (Merge)** : Combinaison des modifications

Histoire

- Années 1970 : Apparition des premiers systèmes de gestion de versions
- **SCCS (Source Code Control System)** : Créé par Bell Labs en 1972
- **RCS (Revision Control System)** : Développé par Walter F. Tichy en 1982
- **CVS (Concurrent Versions System)** : Lancé en 1990, facilitant le travail collaboratif
- **Subversion (SVN)** : Lancé en 2000, offrant des améliorations par rapport à CVS
- **Git : Créé par Linus Torvalds en 2005, devenant le système de gestion de versions décentralisé le plus utilisé**
- **Mercurial** : Également lancé en 2005, offrant une alternative à Git

INSA ROUEN NORMANDIE

Logiciel de gestion de versions 2 / 2

Centralisés vs Décentralisés

- **Centralisés** : Un serveur central stocke toutes les versions (ex : CVS, Subversion)
- **Décentralisés** : Chaque utilisateur possède une copie complète de l'historique (ex : Git, Mercurial)

En résumé

- Facilite la collaboration entre développeurs
- Permet de revenir à des versions antérieures du code
- Aide à gérer les différentes versions du logiciel

INSA ROUEN NORMANDIE

Git en quelques mots 1 / 2

- Créé par Linus Torvalds en 2005
- Conçu pour le développement du noyau Linux
- Système de gestion de versions décentralisé

Particularités

- **Décentralisation** : Chaque utilisateur possède une copie complète de l'historique du projet
- **Efficacité et rapidité** : Conçu pour gérer de grands projets rapidement
- **Intégrité des données** : Utilise des sommes de contrôle (SHA-1) pour chaque fichier et commit
- **Branches légères** : Les branches sont faciles à créer et à gérer

INSA ROUEN NORMANDIE

Git en quelques mots 2 / 2

Principes

- **Snapshots, pas différences** : Git enregistre des snapshots complets de l'état du projet
- **Historique immuable** : Une fois commité, l'historique ne change pas
- **Commit local, push global** : Les modifications sont commitées localement avant d'être poussées vers un dépôt distant
- **Branches et fusions** : Support puissant pour les branches et les fusions, facilitant le développement parallèle

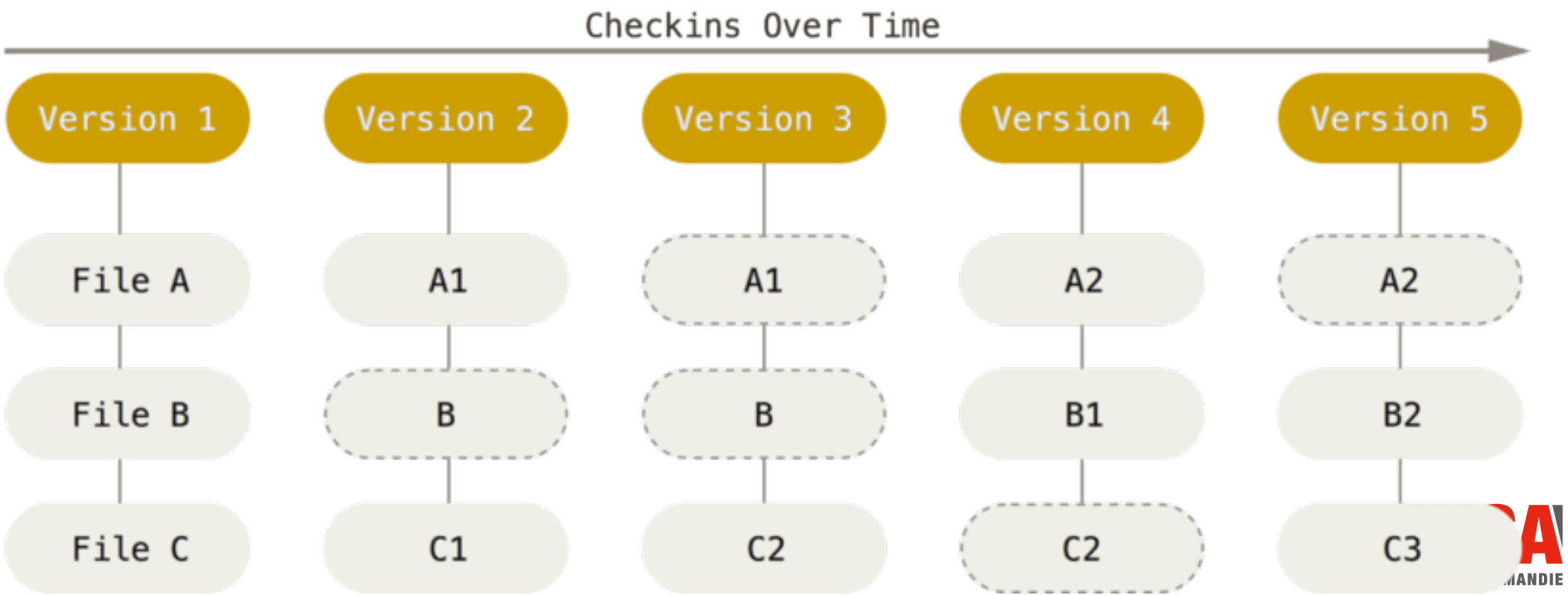
En résumé

- Facilite la collaboration et le développement parallèle
- Permet de maintenir un historique détaillé des modifications
- Assure l'intégrité des données du projet

Les instantanés (*snapshots*)

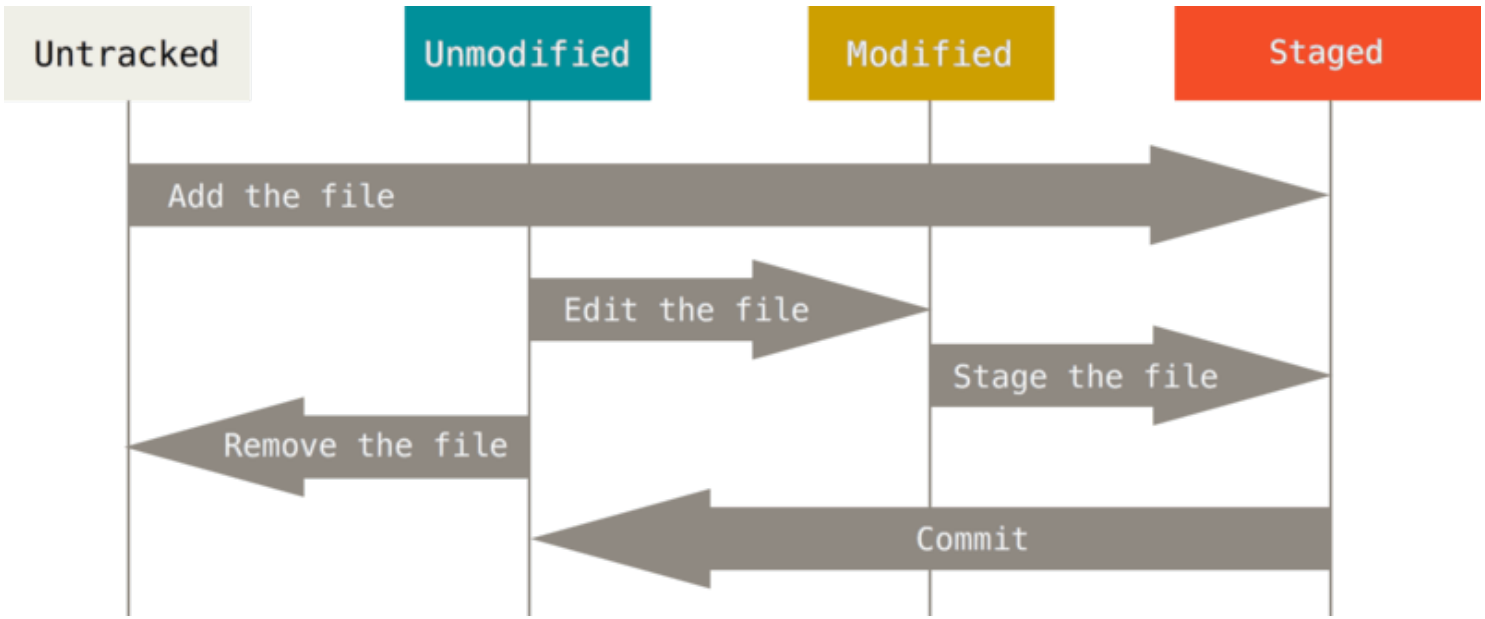
Principe

- On passe d'un instantané à un autre à l'issue d'un *commit*
- Un nouvel instantané est une copie/mise à jour de l'instantané précédent (pas de calcul de différences)
- Tout est local



Les quatre états d'un fichier d'un projet

- **Non suivi** : le fichier n'est pas pris en compte par git
- **Indexé** (*staged*) : le fichier est suivi par git, il sera pris en compte dans le prochain commit
- **Modifié** : le fichier est suivi par git mais a été modifé par rapport au dernier instantané
- **Non modifié** : le fichier est suivi par git et n'a pas été modifié depuis le dernier instantannée



INSA ROUEN NORMANDIE

La commande git 1 / 5

- On utilise depuis le terminal, dans le répertoire du projet, la commande git suivie comme paramètre obligatoire de l'action

Créer d'un projet

`git init [<repertoire>]`

```
$ git init
astuce: Utilisation de 'master' comme nom de la branche initiale. Le nom de la branche
astuce: par défaut peut changer. Pour configurer le nom de la branche initiale
astuce: pour tous les nouveaux dépôts, et supprimer cet avertissement, lancez :
astuce:
astuce:  git config --global init.defaultBranch <nom>
astuce:
astuce: Les noms les plus utilisés à la place de 'master' sont 'main', 'trunk' et
astuce: 'development'. La branche nouvellement créée peut être rénommée avec :
astuce:
astuce:  git branch -m <nom>
Dépôt Git vide initialisé dans /home/delestre/.../exemples/.git/
```

Les branches feront l'objet d'un cours en ITI4

La commande git 2 / 5

Passer un fichier à l'état *indexé*

```
git add nomFichier1 [nomFichier2 ...]
```

```
$ touch main.tex # création d'un fichier vide
$ emacs main.tex # édition du fichier LaTeX
$ pdflatex main.tex # compilation du fichier LaTeX
...
$ ls
main.aux  main.log  main.pdf  main.tex
$ git add main.tex
```

- On n'indexe pas les fichiers créés par une compilation
- On indexe un ou plusieurs fichiers lorsque les modifications forment un tout

INSA ROUEN NORMANDIE

La commande git 3 / 5

Connaître l'état des fichiers

git status

```
$ git status
Sur la branche master

Aucun commit

Modifications qui seront validées :
  (utilisez "git rm --cached <fichier>..." pour désindexer)
nouveau fichier : main.tex

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

Le fichier .gitignore

- Il est possible d'indiquer à Git d'ignorer certains fichiers par l'utilisation du fichier *.gitignore*
- Le site `https://github.com/github/gitignore` propose des *gitignore* pour des langages, éditeurs, etc.

La commande git 4 / 5

Créer un nouvel instantané

```
git commit -m "Message"
```

- Les fichiers indexés passent à l'état *non modifié*
- On réalise un *commit* lorsque les fichiers indexés correspondent à la réalisation d'une tâche unitaire (une nouvelle fonctionnalité, une correction d'un bug, etc.)
- On verra plus tard comment bien rédiger les messages

```
$ git commit -m "Création du fichier principal"
[master (commit racine) fc853b8] Création du fichier principal
 1 file changed, 8 insertions(+)
 create mode 100644 main.tex
$ git status
Sur la branche master
Fichiers non suivis:
  (utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)
main.aux
main.log
main.pdf
```

La commande git 5 / 5

Passer un fichier à l'état *non suivi*

```
git rm <fichier>
```

- Par défaut supprime aussi le fichier
- Pour le passer le fichier à l'état *non suivi* sans le supprimer il faut utiliser l'option `--cached`

```
$ git rm main.tex
rm 'main.tex'
$ ls
main.aux  main.log  main.pdf
$ git status
Sur la branche master
Modifications qui seront validées :
  (utilisez "git restore --staged <fichier>..." pour désindexer)
supprimé :        main.tex

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

Un projet partagé

- Pour l'instant le projet git est uniquement local
- Cela permet une tracabilité du développement mais ca ne permet pas le développement à plusieurs

$\Rightarrow$ Il faut utiliser un serveur (ou dépôt distant)

Rappel

Git est un logiciel de gestion de versions décentralisé : chaque utilisateur possède une copie complète de l'historique des modifications

INSA ROUEN NORMANDIE

La commande git et les dépôts distants 1 / 3

Cloner un dépôt distant

`git clone <URL>`

```
$ git clone git@gitlab.insa-rouen.fr:delestre/cours-iti3-sur-git.git
Clonage dans 'cours-iti3-sur-git'...
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.
$ cd cours-iti3-sur-git/
```

- Cela crée un répertoire du nom du projet
- Deux protocoles de communication sont disponibles SSH et HTTPS

Sur le serveur git de l'INSA, le SSH n'est utilisable qu'au sein de l'INSA

INSA ROUEN NORMANDIE

La commande `git` et les dépôts distants 2 / 3

Récupérer le dernier instantané du dépôt distant

```
git pull
```

- Il peut y avoir un conflit sur un fichier si
  - vous l'avez modifié
  - il a été modifié, *comitté* et *pullé* par un autre développeur après votre dernier `pull`
- Dans ce cas il va falloir résoudre le conflit

INSA ROUEN NORMANDIE

La commande `git` et les dépôts distants 3 / 3

Déposer votre dernier instantané sur le dépôt distant

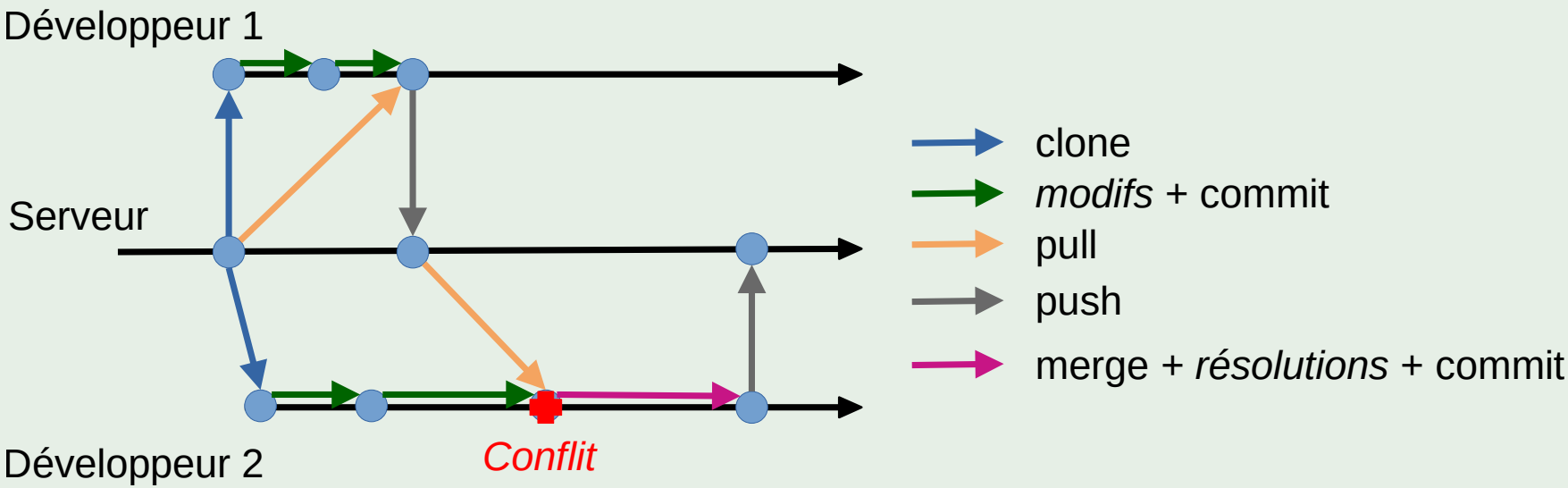
`git push`

On réalise un push lorsqu'on a fini une tâche (un `push` doit être précédé d'un à plusieurs `commit`)

- Pour éviter tout problème, juste avant un `push` il faut faire un `pull`

INSA ROUEN NORMANDIE

Conflit(s) lors d'un `pull` 1 / 2



Conflit(s) lors d'un `pull` 2 / 2

Résolution de conflits

- Il faut dans ce cas éditer les fichiers qui posent problèmes
- Les parties qui posent problèmes sont identifées par :
  - `<<<<<<<` suivi de la partie du fichier local
  - `=======` qui sépare les deux parties
  - `>>>>>>>` qui termine la partie du fichier sur le serveur
- Une fois les résolutions réalisées manuellement, il faut faire un `git add`

INSA ROUEN NORMANDIE

Gitlab

Forge

Une forge permet :

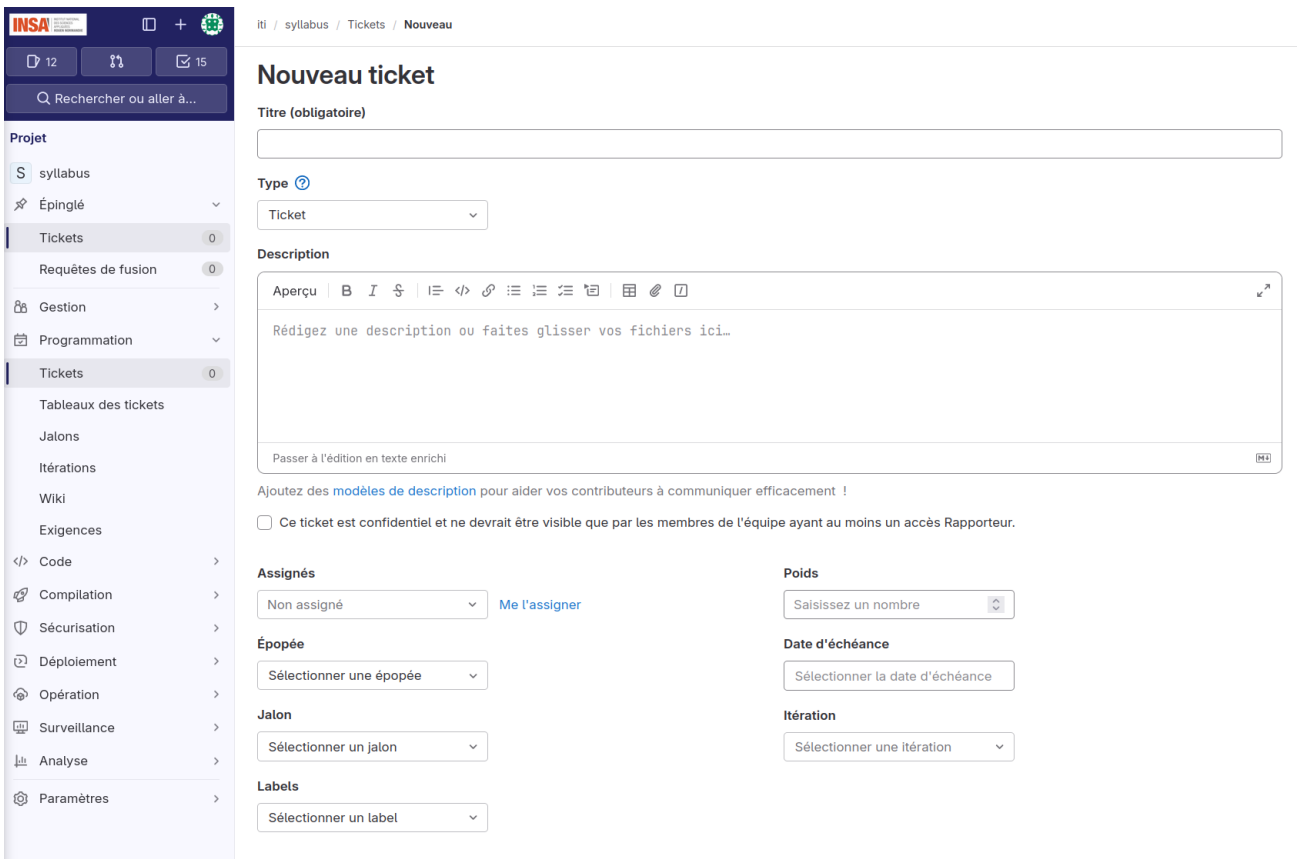
- La gestion de versions : il contient un logiciel de gestion de versions
- Le suivi des bugs et des tâches : concept de tickets
- L'intégration et le déploiement continue : voir ITI4
- La gestion des utilisateurs et de leurs permissions
- L'édition collaborative de la document : wiki

Exemples de Forge

- GitHub
- **GitLab** : `http://gitlab.insa-rouen.fr`
- Bitbucket
- SourceForge

Ticket

- Élément de base du suivi de développement
- Créé pour ajouter une fonctionnalité, corriger un bug : il ne devrait y avoir aucun *commit* non lié à un ticket (le message du *commit* doit y faire référence)



INSA ROUEN NORMANDIE

Conclusion

Nous avons :

- vu ce qu'est un logiciel de gestion de versions et une forge
- vu le b.a.ba de l'utilisation du logiciel de versions git, les commandes principales : `init`, `clone`, `add`, `commit`, `pull`, `push`
- entre aperçu GitLab

Nous ferrons prochainement un TP LaTeX et GitLab

INSA ROUEN NORMANDIE