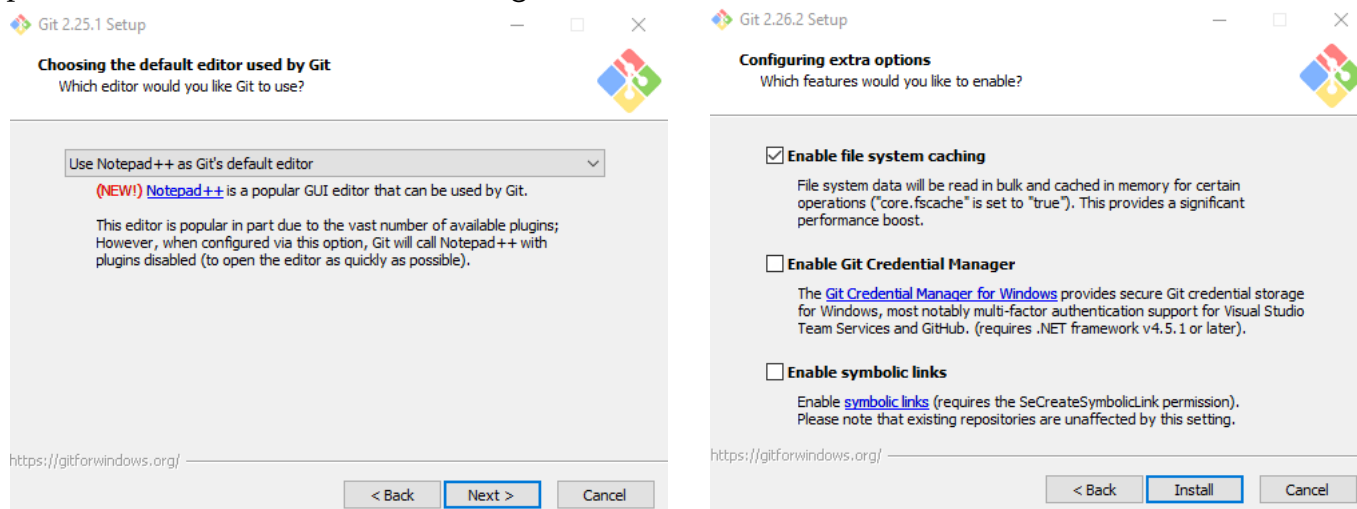


I. Installations et inscriptions

D'abord, installer git : <https://git-scm.com/download>.

On peut laisser toutes les options par défaut, sauf l'éditeur par défaut (sélectionner Notepad++) et décocher l'option Enable Git Credential Manager :



Ensuite, aller sur [gitea](https://gitea.io) (accessible depuis le menu Services du [site de la classe](#)) et cliquer sur S'inscrire en haut à droite.

Mettez un nom d'utilisateur reconnaissable : on pourra être amené à récupérer vos travaux ici.

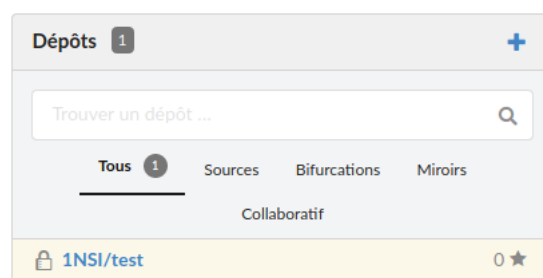
II. Premiers essais : le dépôt test

1) Récupération du dépôt

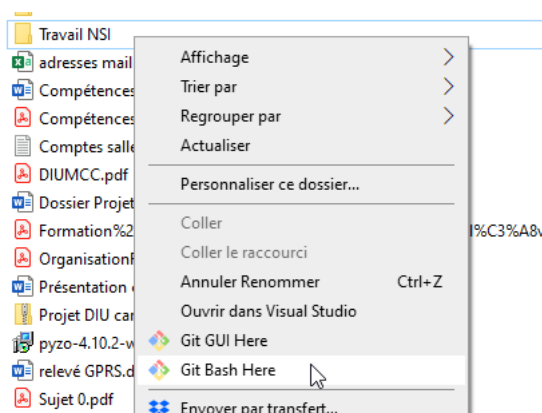
Dès que le prof vous aura ajouté au groupe 1NSI, vous pourrez accéder au dépôt test. Cliquez sur le lien 1NSI/test pour accéder au dépôt.

Vous pouvez alors copier l'adresse du site en cliquant sur le bouton copier :

[HTTPS tans.ddns.info/gitea/1NSI/test.git](https://tans.ddns.info/gitea/1NSI/test.git)

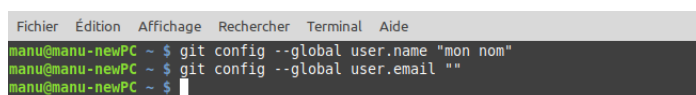


Ensuite, créez un dossier sur votre ordinateur (par exemple Mes Documents/NSI, faites un clic droit dessus, et choisissez Git Bash Here :



Vous devriez avoir une console qui s'ouvre.

Commencez par configurer git en indiquant votre nom et votre adresse mail. Vous pouvez laisser l'adresse mail vide, mais il faut quand même le préciser. Cette étape n'est à faire qu'une fois, sauf si vous avez besoin de changer votre nom d'utilisateur ou votre adresse mail.



Vous pouvez alors récupérer le dépôt sur votre machine en tapant `git clone`, puis en collant l'adresse récupérée plus haut. Comme le dépôt est privé, on vous demande de vous identifier. Tapez les noms d'utilisateur et mot de passe que vous avez créés sur gitea. (*remarque : rien ne s'affiche quand vous tapez votre mot de passe. C'est normal, c'est pour éviter que quelqu'un puisse connaître sa longueur juste en regardant l'écran.*)

```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
manu@manu-newPC ~ $ git clone https://paulconstans.ddns.info/gitea/INSI/test.git
Clonage dans 'test'...
Username for 'https://paulconstans.ddns.info': manu
Password for 'https://manu@paulconstans.ddns.info':
remote: Counting objects: 35, done.
remote: Compressing objects: 100% (23/23), done.
remote: Total 35 (delta 4), reused 0 (delta 0)
Dépaquetage des objets: 100% (35/35), fait.
manu@manu-newPC ~ $ cd test/
manu@manu-newPC ~/test $ ls
bingo  ess  foo.php  pipo  README.md
manu@manu-newPC ~/test $
```

Allez ensuite dans le dossier récupéré (`cd test`), et affichez le contenu (`ls`).

Cette étape n'est nécessaire qu'une fois pour effectuer un clone du dépôt sur votre ordinateur. Une fois que c'est fait, vous pourrez récupérer le contenu du dépôt distant (celui sur gitea) vers le dépôt local (celui sur votre ordinateur) en tapant `git pull`, et envoyer le contenu du dépôt local vers le dépôt distant en tapant `git push`.

2) Quelques commits

Créez un nouveau fichier dans ce dossier (sois en ligne de commande si vous vous rappelez comment faire, soit sous windows).

C'est le moment de la première utilisation d'une commande qui va souvent vous servir quand vous aurez un doute : `git status`

Cette commande vous indique en gros tout ce que vous avez fait depuis la dernière « sauvegarde locale » (on parle de commit). Ici, elle indique la présence d'un nouveau fichier (`fic.txt`) non suivi et

```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
manu@manu-newPC ~/test $ git status
Sur la branche master
Votre branche est à jour avec 'origin/master'.

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

    fic.txt

aucune modification ajoutée à la validation mais des fichiers non suivis sont pr
ésents (utilisez "git add" pour les suivre)
manu@manu-newPC ~/test $
```

elle donne la commande à effectuer pour l'ajouter aux fichiers suivis : `git add`.

On obtempère donc, et on regarde ce que donne maintenant `git status` :

```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
manu@manu-newPC ~/test $ git add fic.txt
manu@manu-newPC ~/test $ git status
Sur la branche master
Votre branche est à jour avec 'origin/master'.

Modifications qui seront validées :
  (utilisez "git reset HEAD <fichier>..." pour désindexer)

    nouveau fichier : fic.txt

manu@manu-newPC ~/test $
```

On est donc prêt pour le premier commit. Si on tape juste `git commit`, Notepad++ va s'ouvrir (si vous avez bien changé l'éditeur par défaut au moment de l'installation) avec un rappel des informations concernant le commit, et vous demandant d'écrire un message de validation :

```
# Veuillez saisir le message de validation pour vos modifications. Les lignes
# commençant par '#' seront ignorées, et un message vide abandonne la validation
#
# Sur la branche master
# Votre branche est à jour avec 'origin/master'.
#
# Modifications qui seront validées :
#    nouveau fichier : fic.txt
#
```

On retrouve dans les commentaires le message renvoyé par `git status`.

Tapez le message que vous voulez (genre « ajout de mon premier fichier sur gitea »), enregistrez et quittez.

```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
manu@manu-newPC ~/test $ git commit
[master fa84ec6] ajout de mon premier fichier sur gitea
1 file changed, 1 insertion(+)
create mode 100644 fic.txt
manu@manu-newPC ~/test $ git status
Sur la branche master
Votre branche est en avance sur 'origin/master' de 1 commit.
(utilisez "git push" pour publier vos commits locaux)

rien à valider, la copie de travail est propre
manu@manu-newPC ~/test $
```

On voit maintenant que tout le travail effectué à été sauvegardé sur notre ordinateur, mais qu'on a fait un commit depuis la dernière synchronisation avec le serveur.

Avant de synchroniser, modifiez le fichier que vous avez créé (ajouter un ligne, changer un mot,) puis affichez le status :

```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
manu@manu-newPC ~/test $ git status
Sur la branche master
Votre branche est en avance sur 'origin/master' de 1 commit.
(utilisez "git push" pour publier vos commits locaux)

Modifications qui ne seront pas validées :
  (utilisez "git add <fichier>..." pour mettre à jour ce qui sera validé)
  (utilisez "git checkout -- <fichier>..." pour annuler les modifications dans l
a copie de travail)

      modifié :      fic.txt

aucune modification n'a été ajoutée à la validation (utilisez "git add" ou "git
commit -a")
manu@manu-newPC ~/test $
```

Le fichier fic.txt étant déjà indexé, on peut se passer de l'étape `git add fic.txt` en disant qu'on veut actualiser tous les fichiers modifiés à l'aide de l'option `-a` de `commit`. De plus, on peut donner directement un message de validation avec l'option `-m`. On a donc :

```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
manu@manu-newPC ~/test $ git commit -a -m "Déjà mon deuxième commit. Le temps pa
sse..."
[master e4f2e6b] Déjà mon deuxième commit. Le temps passe...
1 file changed, 1 insertion(+)
manu@manu-newPC ~/test $ git status
Sur la branche master
Votre branche est en avance sur 'origin/master' de 2 commits.
(utilisez "git push" pour publier vos commits locaux)

rien à valider, la copie de travail est propre
manu@manu-newPC ~/test $
```

3) Mise à jour sur gitea

Tout le travail effectué jusqu'à présent sur le dépôt s'est fait en local (sur votre ordinateur). Il faut maintenant actualiser sur gitea. Dans l'idéal, si personne d'autre n'a travaillé sur le dépôt en même temps que vous, il suffit de taper `git push` et de renseigner vos noms d'utilisateur et mot de passe sur gitea.

```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
manu@manu-newPC ~/test $ git push
Username for 'https://paulconstans.ddns.info': prof
Password for 'https://prof@paulconstans.ddns.info':
Décompte des objets: 6, fait.
Delta compression using up to 4 threads.
Compression des objets: 100% (5/5), fait.
Écriture des objets: 100% (6/6), 564 bytes | 188.00 KiB/s, fait.
Total 6 (delta 2), reused 0 (delta 0)
To https://paulconstans.ddns.info/gitea/1NSI/test.git
a5f9328..e4f2e6b master -> master
manu@manu-newPC ~/test $
```

Remarque : Le Raspberry sur lequel est installé gitea n'est pas très puissant. La mise à jour peut donc prendre un moment (jusqu'à un trentaine de seconde).

4) Gestion des conflits

Il est possible que quelqu'un d'autre ait travaillé sur le dépôt en même temps que vous. Dans ce cas, au moment de faire le push, vous allez avoir un message vous conseillant de refaire un pull.

```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
manu@manu-newPC ~/test $ git push
Username for 'https://paulconstans.ddns.info': manu
Password for 'https://manu@paulconstans.ddns.info':
To https://paulconstans.ddns.info/gitea/1NSI/test.git
 ! [rejected]        master -> master (fetch first)
error: impossible de pousser des références vers 'https://paulconstans.ddns.info/gitea/1NSI/test.git'
astuce: Les mises à jour ont été rejetées car la branche distante contient du travail que
astuce: vous n'avez pas en local. Ceci est généralement causé par un autre dépôt poussé
astuce: vers la même référence. Vous pourriez intégrer d'abord les changements distants
astuce: (par exemple 'git pull ...') avant de pousser à nouveau.
astuce: Voir la 'Note à propos des avancées rapides' dans 'git push --help' pour plus d'information.
manu@manu-newPC ~/test $
```

Si le travail effectué en parallèle du votre s'est effectué sur des fichiers sur lesquels vous n'avez pas travaillé, ou éventuellement sur les mêmes fichiers mais sur des parties différentes, il suffit de commencer par faire un `git pull` puis de refaire un `git push`.

Mais il est aussi possible que quelqu'un ait travaillé sur la même partie du fichier que vous, auquel cas la fusion automatique n'est pas possible, et il va falloir un peu de travail :

```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
manu@manu-newPC ~/test $ git pull
Username for 'https://paulconstans.ddns.info': manu
Password for 'https://manu@paulconstans.ddns.info':
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0)
Dépaquetage des objets: 100% (3/3), fait.
Depuis https://paulconstans.ddns.info/gitea/1NSI/test
 e4f2e6b..ca5edec  master    -> origin/master
Fusion automatique de pipo
CONFLIT (contenu) : Conflit de fusion dans pipo
La fusion automatique a échoué ; réglez les conflits et validez le résultat.
manu@manu-newPC ~/test $
```

Ici, il y a un problème dans le fichier `pipo`. On doit donc l'ouvrir (avec un éditeur quelconque, par exemple Notepad++) pour corriger les problèmes. Dans le fichier, les zones en conflit sont de la forme :

```
<<<<<<< HEAD
contenu du fichier local en conflit
=====
contenu du fichier distant en conflit
>>>>>>> un nombre compliqué en hexadécimal
```

Vous devez remplacer toute la zone par la modification choisie (ça peut être le contenu d'une des deux fichiers, ou un mélange des deux, ou n'importe quoi d'autre).

Une fois le fichier modifié et sauvegardé, taper `git add` suivi du nom de fichier corrigé, puis `git commit` pour valider la fusion. On peut alors finaliser le push.

```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
manu@manu-newPC ~/test $ git add pipo
manu@manu-newPC ~/test $ git commit
[master 2cc7d97] Merge branch 'master' of https://paulconstans.ddns.info/gitea/1NSI/test
manu@manu-newPC ~/test $ git push
Username for 'https://paulconstans.ddns.info': manu
Password for 'https://manu@paulconstans.ddns.info':
Décompte des objets: 6, fait.
Delta compression using up to 4 threads.
Compression des objets: 100% (5/5), fait.
Écriture des objets: 100% (6/6), 594 bytes | 594.00 KiB/s, fait.
Total 6 (delta 2), reused 0 (delta 0)
To https://paulconstans.ddns.info/gitea/1NSI/test.git
 ca5edec..2cc7d97  master    -> master
manu@manu-newPC ~/test $
```

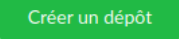
III. Créer votre propre dépôt

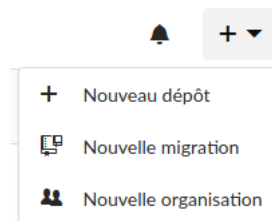
1) Création

Pour créer un nouveau dépôt sur gitea, cliquer sur le + en haut à droite, et choisissez Nouveau dépôt, ou alors cliquez sur + à droite de l'onglet Dépôts.


Choisissez un nom pour votre dépôt, sans espace ni caractère spéciaux, sauf éventuellement des tirets ("-"), des tirets bas ("_"), et des points("."). Vous pouvez éventuellement ajouter une description.

Si vous ne voulez pas que les autres voient votre dépôt, vous pouvez cocher l'option Rendre le dépôt privé. Vous pouvez aussi ajouter un fichier LISEZMOI en cochant Initialiser le dépôt.

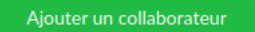
Cliquez ensuite sur le bouton . Vous pouvez alors travailler avec ce dépôt comme vu précédemment avec le dépôt test.



2) Paramètres

Pour modifier votre dépôt, vous disposez du bouton  Paramètres.

L'onglet Dépôt permet entre autre de modifier le nom de votre dépôt, sa visibilité, voire de le supprimer (attention c'est irréversible).

Un autre onglet qui vous sera probablement utile est l'onglet Collaborateurs dans lequel vous allez pouvoir ajouter des collaborateurs en tapant quelques lettres de son login, en le sélectionnant dans le menu déroulant et en cliquant sur . Chaque collaborateur peut avoir un des trois accès :

Lecture : peut uniquement lire votre dépôt (en ligne, ou grâce a `git clone` et `git push`). Cette option est utile uniquement si vous avez déclaré votre dépôt privé, sinon tout le monde peut lire son contenu.

Écriture : peut lire votre dépôt ou écrire dedans (en ligne ou grâce à `git push` après l'avoir cloné).

Administrateur : a tous les droits sur le dépôt, comme le renommer, changer sa visibilité, ajouter ou supprimer des collaborateurs, etc.

IV. Gestion des branches

Le principe des branches permet de travailler sur plusieurs versions de votre code en parallèle. Supposons par exemple que vous aillez une super idée pour votre programme, mais qui demande de modifier des morceaux un peu compliqués qui marchent pour l'instant bien. Vous ne voulez peut-être pas tout de suite mettre le bazar dans ce qui marche avant d'être sûr que votre super idée va aboutir. Pour ça, on commence par créer une nouvelle branche (`git branch`), puis on se place dans cette branche (`git checkout`) :

```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
manu@manu-newPC ~/test $ git branch maSuperBranche
manu@manu-newPC ~/test $ git checkout maSuperBranche
Basculement sur la branche 'maSuperBranche'
manu@manu-newPC ~/test $
```

On peut alors effectuer toute les modifications nécessaires, et effectuer des commits. Tout se passe dans la nouvelle branche sans modifier la branche principale. Si on veut retourner dans cette dernière, il suffit de taper `git checkout master`. De façon générale, on passe d'une branche à une autre par le commande `git checkout nom_de_la_branche`. On peut avoir la liste des branches existantes avec `git branch`. La branche dans laquelle on se trouve est précédée d'une étoile :

```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
manu@manu-newPC ~/test $ git branch
* maSuperBranche
  master
manu@manu-newPC ~/test $ git checkout master
Basculement sur la branche 'master'
Votre branche est à jour avec 'origin/master'.
manu@manu-newPC ~/test $ git branch
  maSuperBranche
* master
manu@manu-newPC ~/test $
```

Vous pouvez alors vérifier que vous retrouvez tous les fichiers exactement dans l'état où ils étaient avant de créer la branche.

Par défaut, les branches sont définies juste en local. Le but est souvent plutôt de tester une idée. Si on veut quand même partager la branche et qu'on essaye de faire un push, on est averti par un message, avec la méthode à suivre pour pousser effectivement la branche :

```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
manu@manu-newPC ~/test $ git push
fatal: La branche courante maSuperBranche n'a pas de branche amont.
Pour pousser la branche courante et définir la distante comme amont, utilisez

git push --set-upstream origin maSuperBranche

manu@manu-newPC ~/test $ git push --set-upstream origin maSuperBranche
Username for 'https://paulconstans.ddns.info': manu
Password for 'https://manu@paulconstans.ddns.info':
Décompte des objets: 3, fait.
Delta compression using up to 4 threads.
Compression des objets: 100% (2/2), fait.
Écriture des objets: 100% (3/3), 254 bytes | 254.00 KiB/s, fait.
Total 3 (delta 1), reused 0 (delta 0)
To https://paulconstans.ddns.info/gitea/1NSI/test.git
 * [new branch]      maSuperBranche -> maSuperBranche
La branche 'maSuperBranche' est paramétrée pour suivre la branche distante 'maSuperBranche' depuis 'origin'.
manu@manu-newPC ~/test $
```

Enfin, si votre super idée aboutit, vous pouvez intégrer tout le travail dans la branche principale en commençant par se placer dans cette dernière (`git checkout master`), puis en fusionnant votre branche (`git merge`). Si la branche principale n'a pas évoluées depuis la bifurcation, la fusion est immédiate. Sinon, il pourra peut-être y avoir des conflits à gérer, comme on l'a vu au 4).

Une fois la fusion effectuée, on peut (si on veut) supprimer la branche devenue inutile (`git branch -d`).

```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
manu@manu-newPC ~/test $ git merge maSuperBranche
Mise à jour 2cc7d97..8e65ad1
Fast-forward
 maSupermodif | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 maSupermodif
manu@manu-newPC ~/test $ git branch -d maSuperBranche
Branche maSuperBranche supprimée (précédemment 8e65ad1).
manu@manu-newPC ~/test $
```