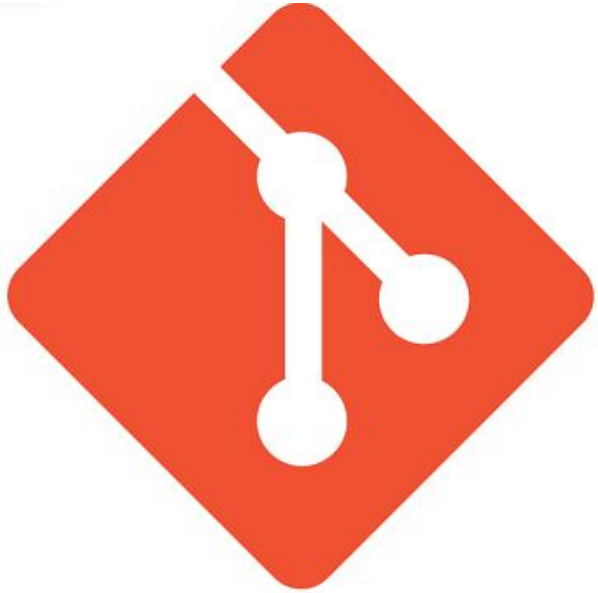
GIT

2021-2022

ESPRIT thouraya.louati@esprit.tn UP ASI (Bureau E204)

Séance 4

- Séquence 4: Mise en place du GIT et test des commandes basiques et avancées
  - Charge : 3 heures
    - Les SCM
      - Savoir l'utilité des SCM
      - Savoir les différentes architectures des SCM
      - Installer et utiliser GIT
      - Définir les techniques de versionning et de Branching
      - Utiliser GIT avec Jenkins
    - Installation GIT.
    - Démo
    - **TP1:** Les commandes GIT basiques
    - **TP2:** Les commandes GIT avancées



git

SCM Version Control Systems

- Les **systèmes de contrôle de version** sont des logiciels qui fournissent des fonctionnalités de contrôle de version (**Git, Subversion, TFS Version Control**).
- **Software Configuration Management** est un terme plus large qui englobe tous les processus nécessaires pour **build, package et deploy** des applications. Ceci inclut les systèmes de contrôle de version.
- **«Distributed VCS (Version Control System)»**

SCM Version Control Systems

- Git est défini comme un outil permettant le **contrôle des versions d'un projet, mono ou multi contributeurs**.
- Il a été créé en 2005 par Linus Torwald pour gérer le code source du noyau Linux, projet dépassant aujourd'hui les 17'000'000 lignes de code et rassemblant généralement plus de 1'200 contributeurs pour chaque release majeure.

SCM Version Control Systems

Voici quelques caractéristiques principales de git :

- **Version Control** -> Chaque étape de développement du projet est mémorisé, comme pris en photo.
- **Distribué** -> Chaque *dépôt*, sur chaque machine est autonome. Pas besoin d'être connecté pour travailler.
- **Collaboratif** -> Lorsque plusieurs personnes souhaitent collaborer sur un même projet, l'utilisation d'un *dépôt* de référence permet de gérer les contributions de chacun.
- *Github* est l'hébergement le plus utilisé pour cela. https://github.com/
- Des alternatives libres existent également, comme *Gitlab* qui propose un hébergement sur leur infrastructure, ou de télécharger la solution pour la déployer sur l'infrastructure d'une entreprise. https://about.gitlab.com/
- **Flexible** -> De tout petits à de très larges projets, git est conçu pour être extensible. De 1 à des milliers de contributeurs, de quelques *commits* à des milliers, d'une *branche* à des centaines … git est conçu pour.

Commandes GIT

- **git-clone** - Clone un dépôt dans un nouveau répertoire.
- **git-init** - Crée un dépôt Git vide ou réinitialise un dépôt existant
- **git-add** - Ajoute le contenu de fichiers à l'index
- **git-mv** - Déplace ou renomme un fichier, un répertoire, ou un lien symbolique
- **git-reset** - Réinitialise la HEAD actuelle à l'état spécifié
- **git-rm** - Supprime des fichiers de l'arbre de travail et de l'index
- **git-branch** - Liste, crée, ou supprime des branches

Commandes GIT

- **git-checkout** - Bascule sur une autre branche ou restaure des fichiers de l'arbre de travail
- **git-commit** - Enregistrer les modifications dans le dépôt
- **git-diff** - Affiche les modifications entre les commits, un commit et l'arbre de travail, etc
- **git-merge** - Fusionne deux ou plusieurs historiques de développement ensemble
- **git-fetch -** Télécharger les objets et références depuis un autre dépôt
- **git-pull** - Rapatrier et intégrer un autre dépôt ou une branche locale

Commandes GIT

- **git-push -** Met à jour les références distantes ainsi que les objets associés
- **git-revert -** Revert some existing commits

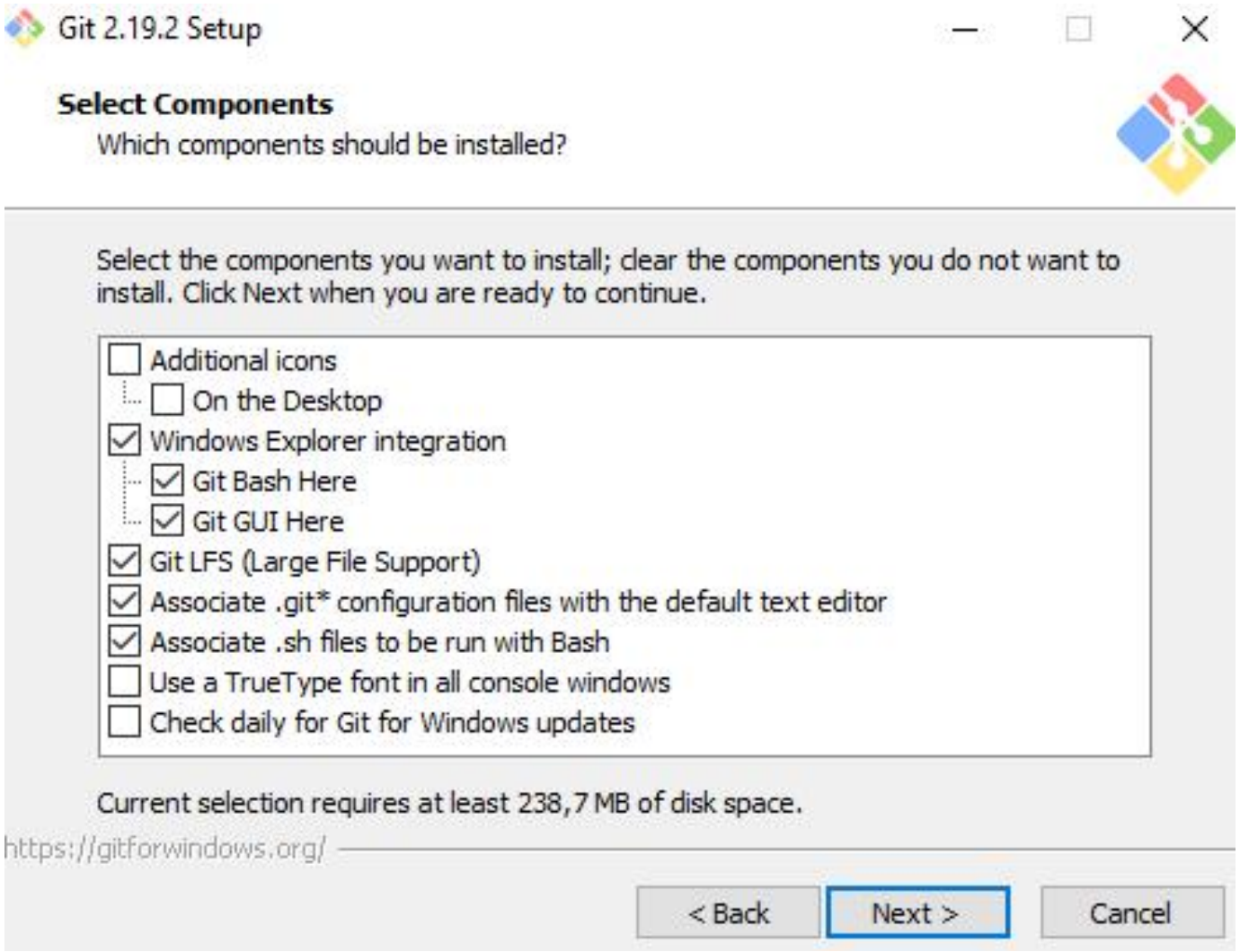
- **Les commandes basiques :** init, clone, fetch, pull, commit, push, merge, branches
- **Les commandes plus avancées :** diff, tag, reset, revert, git-flow

Installation GIT

- Télécharger et exécuter git sur le lien suivant https://git-scm.com/downloads en fonction de votre système d'exploitation

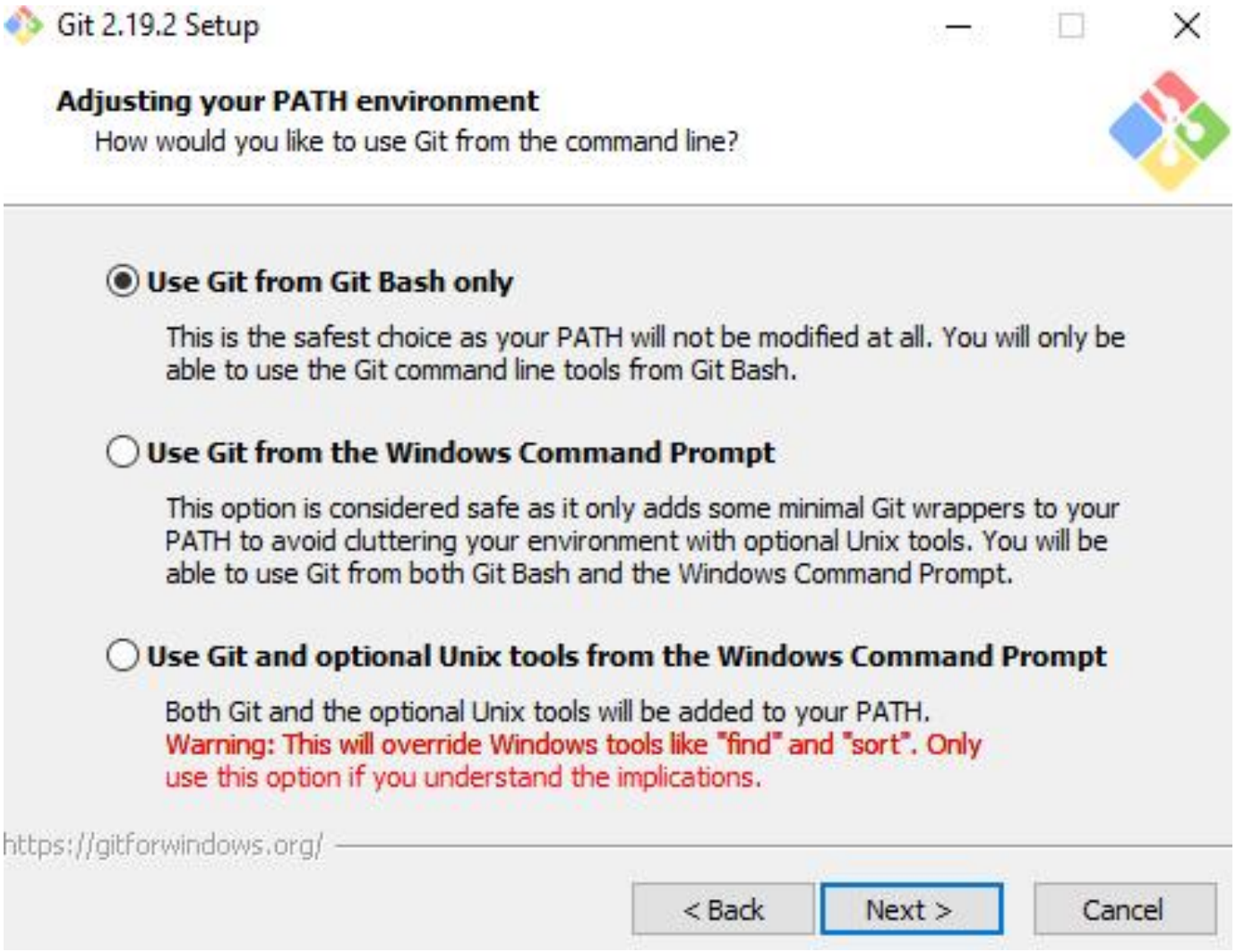
Installation GIT

- Choisissez les options suivantes et cliquer Next.



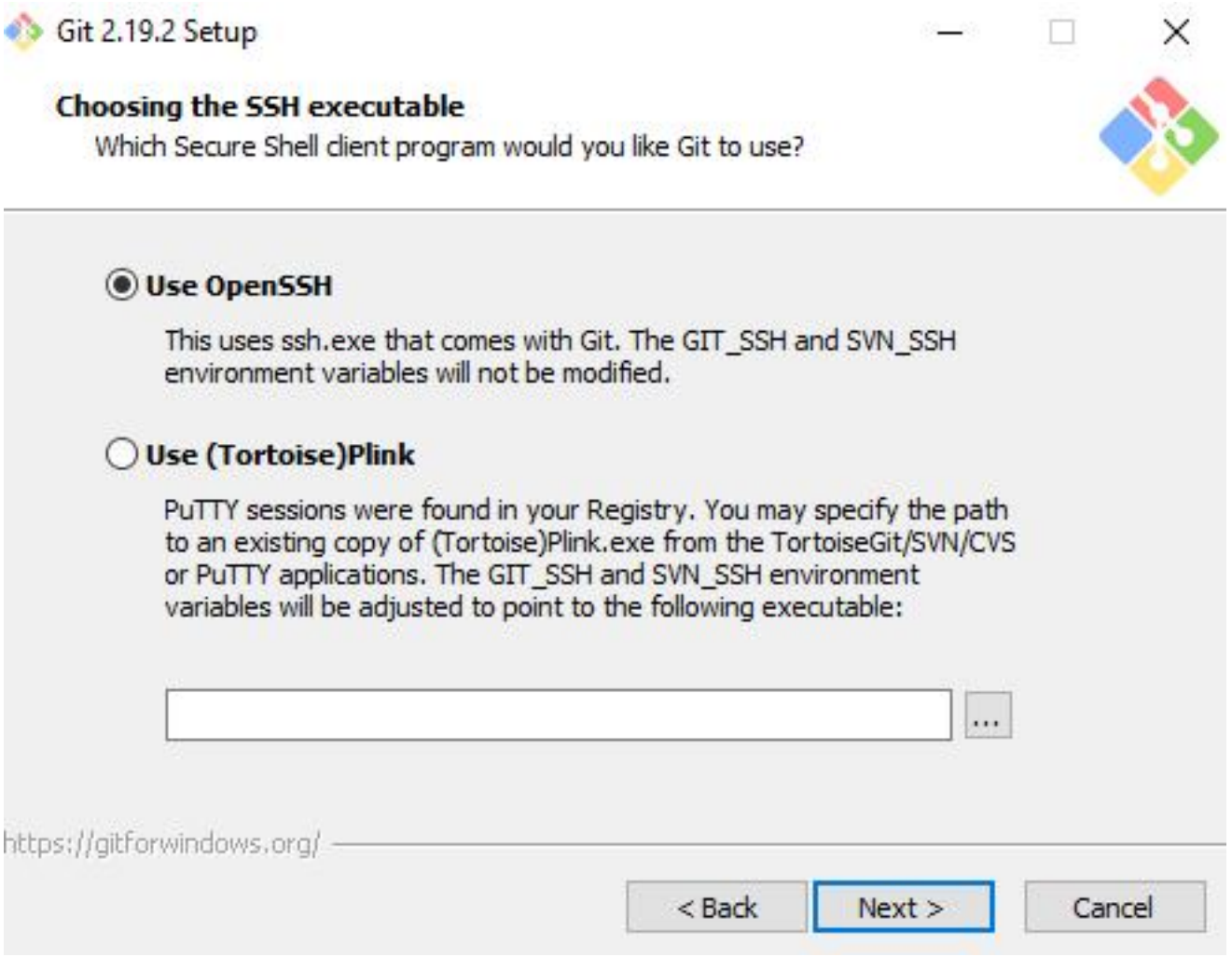
Installation GIT

- Choisissez l'option suivante et cliquer Next.



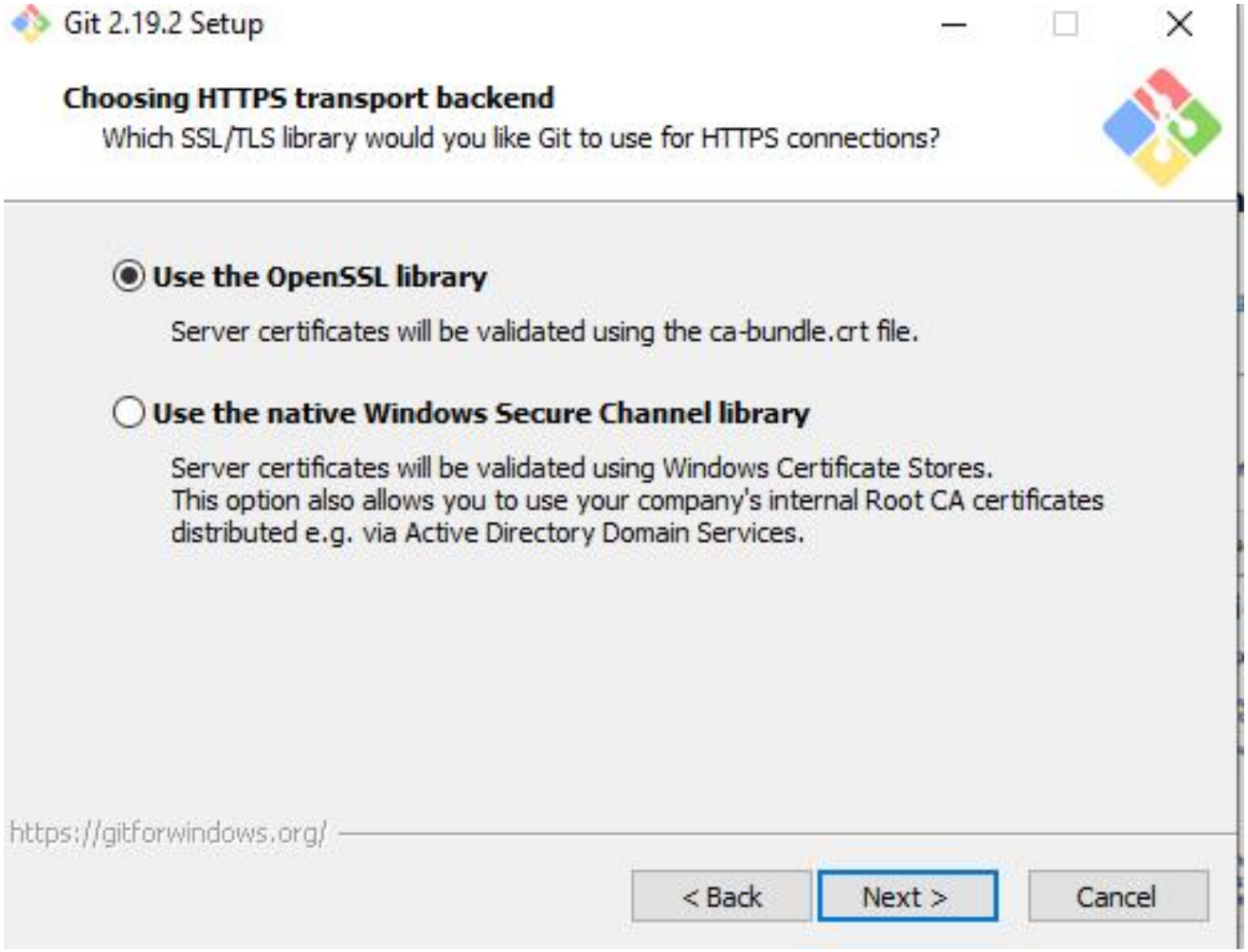
Installation GIT

- Choisissez l'options suivante et cliquer Next.



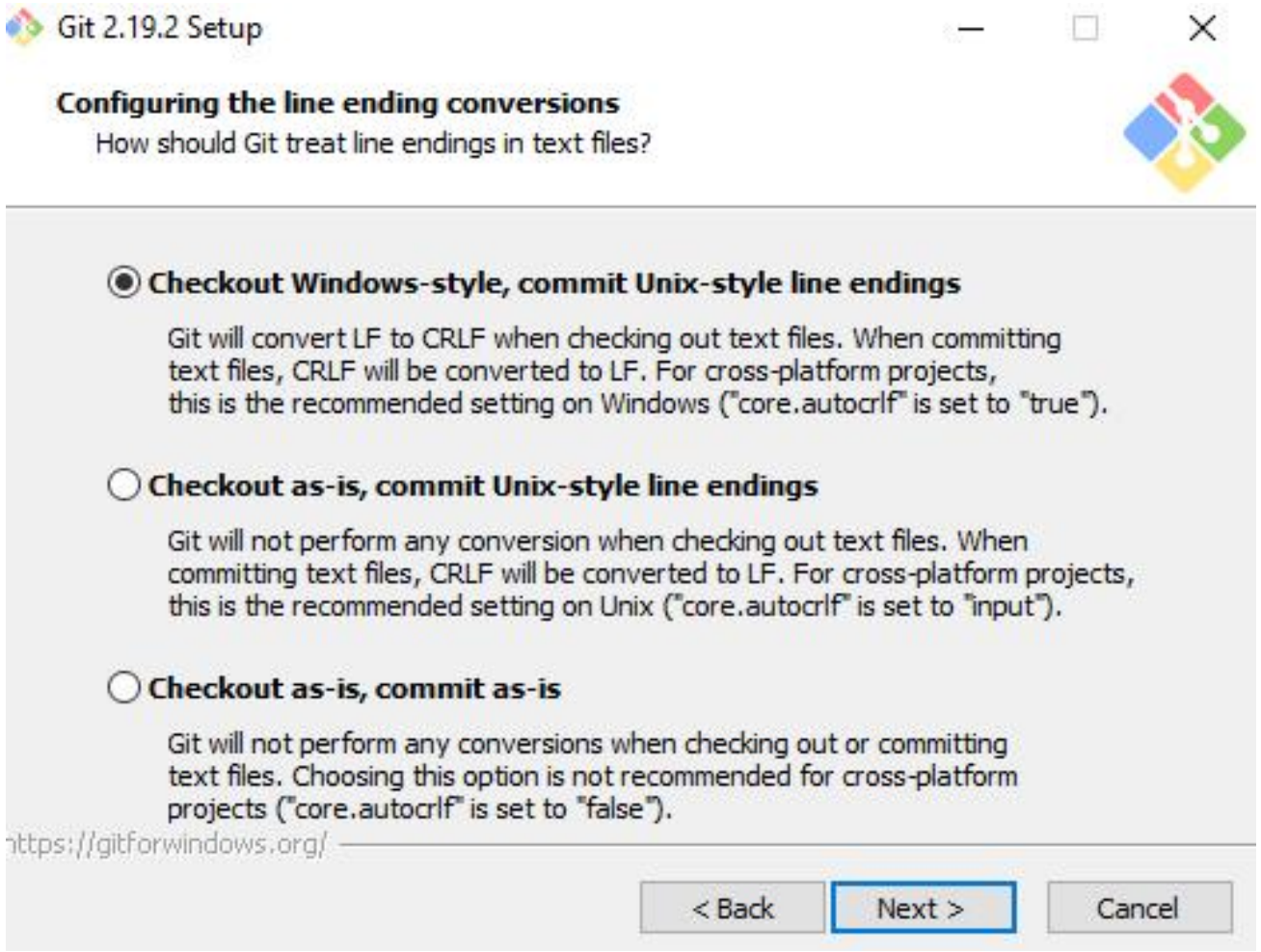
Installation GIT

- Choisissez l'option suivante et cliquer Next.



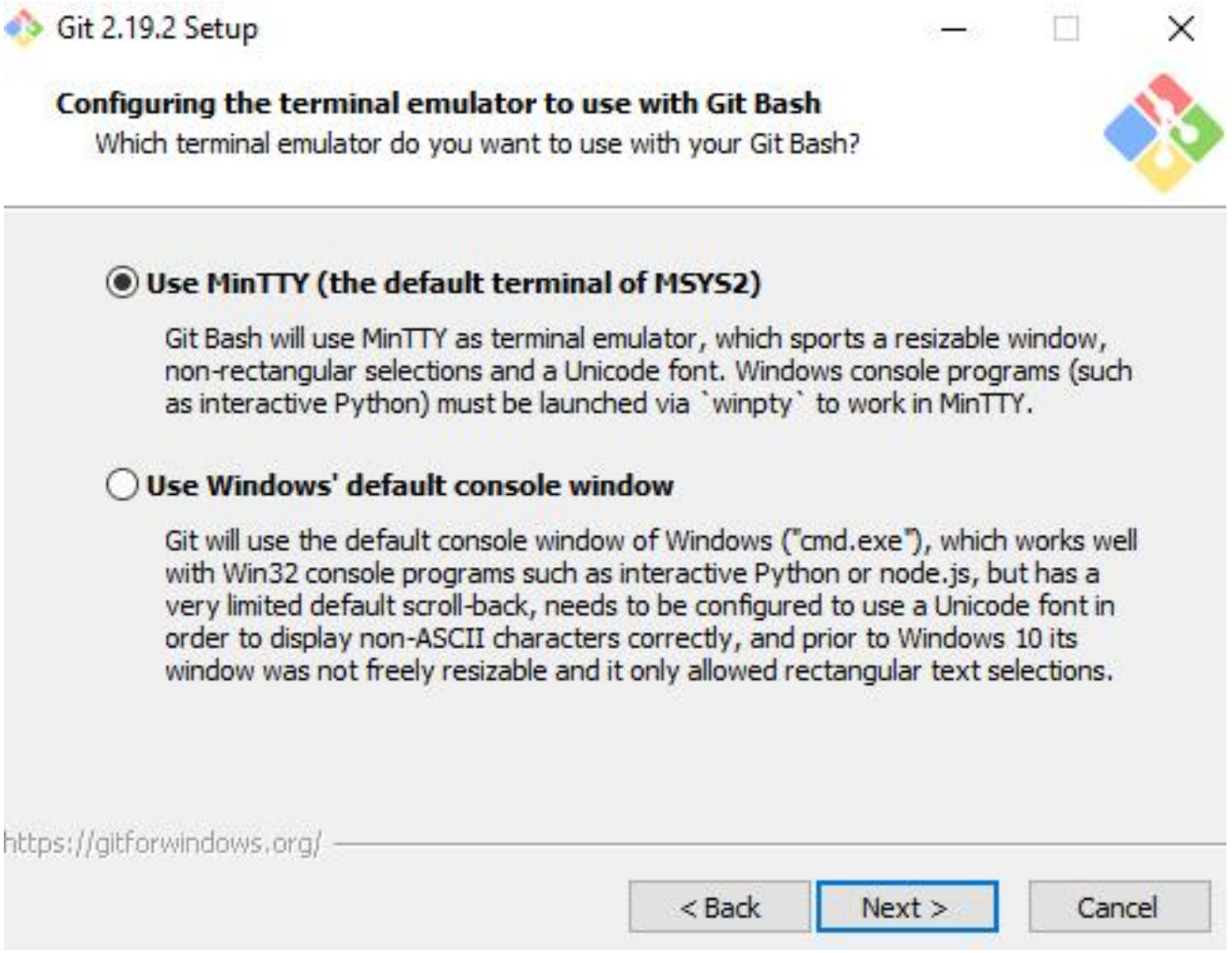
Installation GIT

- Choisissez l'option suivante et cliquer Next.



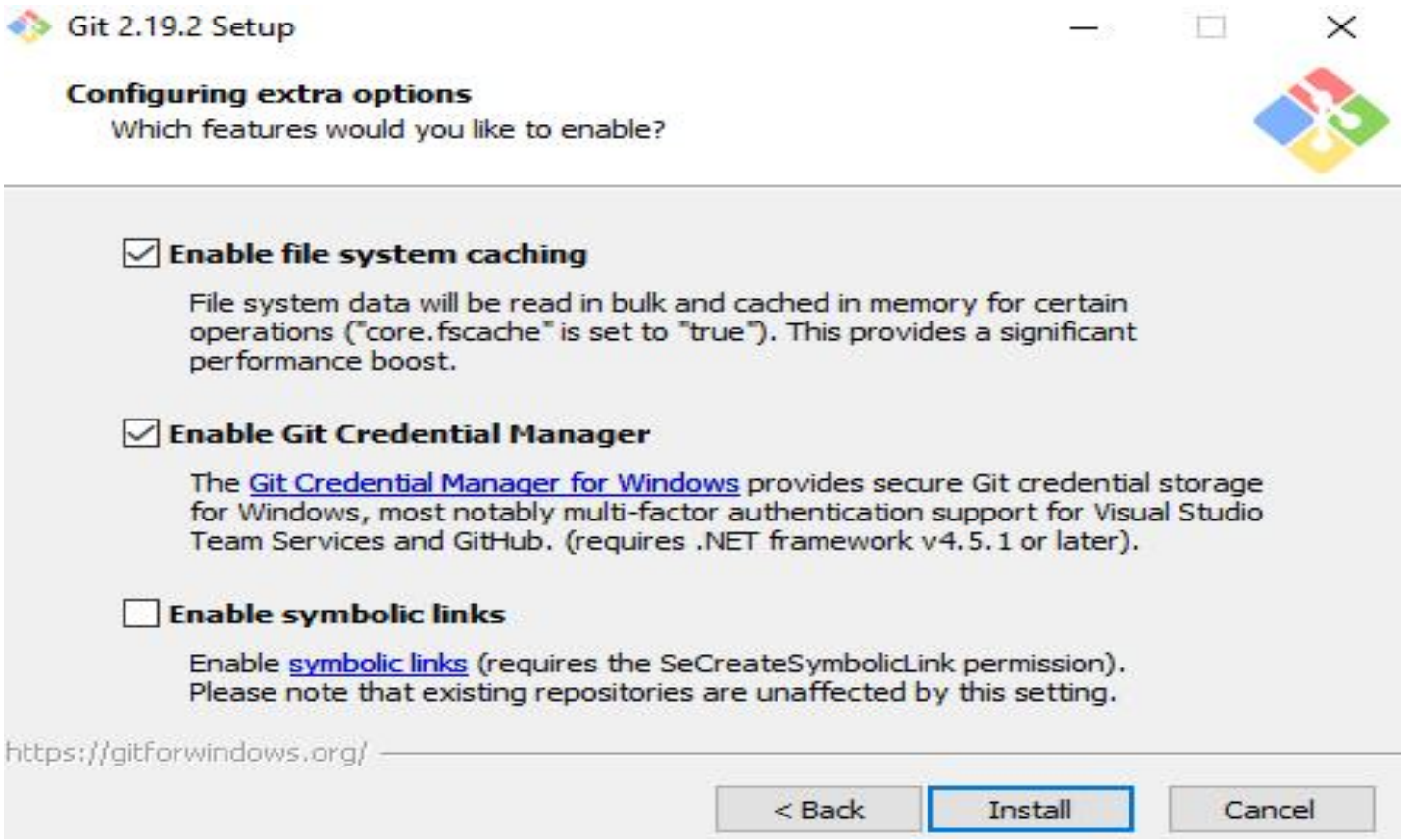
Installation GIT

- Choisissez l'option suivante et cliquer Next.



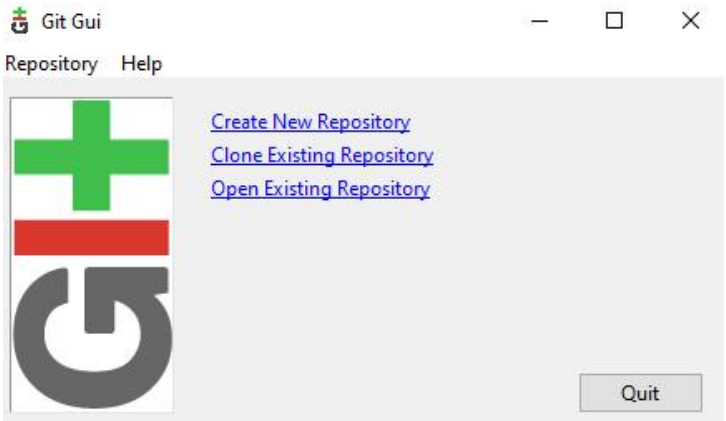
Installation GIT

- Choisir votre répertoire d'installation. Cliquer sur Next en laissant les paramètres par défaut jusqu'à obtenir le bouton Install.



Installation GIT

- Attendre que l'installation s'effectue et cliquer sur Next pour terminer.
- **Mettre à jour la variable d'environnement PATH avec C:\Program Files\Git\bin**
- Lancez une nouvelle commande MS-DOS et exécuter : git-version pour avoir la version de Git installée.
- Vous pouvez travailler, aussi, sur GitGUI



Démo GIT

- Repo :GitHub ou GitLab
  - Scénario 1 : 1 producer, many consumers.
  - Scénario 2 : n producers, 1 consumer (avec et sans conflit)
  - Scénario 3 : branche par développeur

Démo

- Ouvrir Git Bash et lancer git --version pour vérifier que Git est bien installé :

```
ASUS@DESKTOP-41UCFB9 MINGW64 ~
$ git --version
git version 2.19.2.windows.1
```

- Initialisation de GIT:
  - Toujours sur Git Bash, définir votre identité, avec **git config, pour éviter que Git vous demande cela à** chaque action.

```
ASUS@DESKTOP-41UCFB9 MINGW64 ~
$ git config --global user.name "Thouraya LOUATI"

ASUS@DESKTOP-41UCFB9 MINGW64 ~
$ git config --global user.name "thouraya.louati@gmail.com"
```

Démo

- Créer le dossier demodevops.

```
ASUS@DESKTOP-41UCFB9 MINGW64 ~
$ cd Desktop/

ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop
$ mkdir demodevops
```

- Aller dans ce dossier et initialiser Git pour pouvoir l'utiliser sur ce projet (**git init puis git status**) :

```
ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop
$ cd demodevops/

ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demodevops
$ git init
Initialized empty Git repository in C:/Users/ASUS/Desktop/demodevops/.git/

ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demodevops (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

Démo


```
ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demodevops (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

- Créer un fichier file.txt

```
ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demodevops (master)
$ touch file.txt

ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demodevops (master)
$ ls
file.txt
```

Démo

- Sélectionner les fichiers à commiter avec la commande **git add**
  - Puis un **git status**
  - Puis faire un **git commit . Cela revient à dire à Git de**

    prendre une photo de ton projet.

```
ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demodevops (master)
$ git add file.txt

ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demodevops (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   file.txt
```

Démo

- **git commit**

```
ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demodevops (master)
$ git commit -m "initialisation du projet"
[master (root-commit) 8abc0e2] initialisation du projet
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file.txt
```

https://enacit.epfl.ch/cours/git/

- Après chaque modification d'un ou de plusieurs fichiers, vous pouvez refaire la même action pour commiter vos modifications : **git status puis git add** puis **git commit** (avec un autre commentaire bien sûr).

Modifier le contenu du fichier file.txt et refaites les actions ci-dessous :

Démo


```
ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demodevops (master)
$ git status
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:   file.txt

no changes added to commit (use "git add" and/or "git commit -a")

ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demodevops (master)
$ git add file.txt

ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demodevops (master)
$ git commit -m "Modification du fichier file.txt"
[master e3e870e] Modification du fichier file.txt
 1 file changed, 1 insertion(+)
```

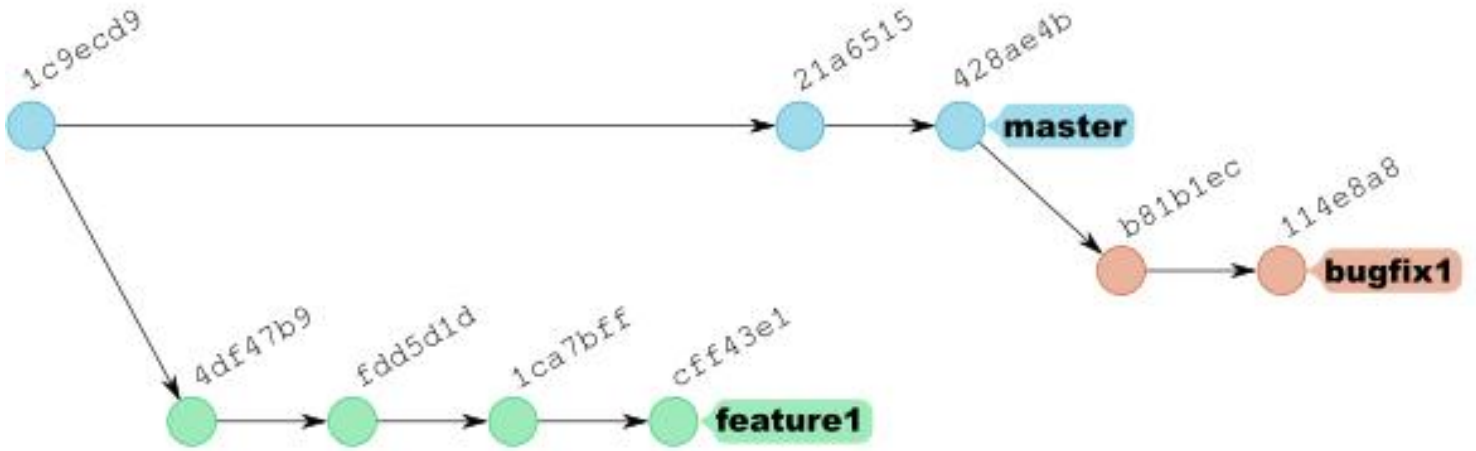
Démo

- Historique.
  - Lancer la commande **git log** pour voir les deux commit (les deux photos de ton projet) :

```
ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demodevops (master)
$ git log
commit e3e870e014a32805bbac1c217f6c528618ecffdb (HEAD -> master)
Author: thouraya.louati@gmail.com <thouraya.louati@esprit.tn>
Date:   Sat Oct 2 17:12:36 2021 +0100

    Modification du fichier file.txt

commit 8abc0e268b4f938142d178c10401633e28737b64
Author: thouraya.louati@gmail.com <thouraya.louati@esprit.tn>
Date:   Sat Oct 2 16:50:48 2021 +0100

    initialisation du projet
```

Démo

- **Les branches.**



  - Créer une branche (**git branch**) et **lister les branches** (le * indique sur quelle branche nous sommes actuellement) :

```
ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demodevops (master)
$ git branch mabranche

ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demodevops (master)
$ git branch
  mabranche
* master
```

Démo

- Basculer sur la branche mabranche.

```
ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demodevops (master)
$ git checkout mabranche
Switched to branch 'mabranche'

ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demodevops (mabranche)
$ git branch
* mabranche
  master
```

Démo

- Faite une modification sur le fichier file.txt et commiter une modification sur cette branche :

```
ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demodevops (mabranche)
$ git add file.txt

ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demodevops (mabranche)
$ git commit -m "Modification 3 du fichier file.txt"
[mabranche 996ecfa] Modification 3 du fichier file.txt
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Démo


```
ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demodevops (mabranche)
$ git log
commit 996ecfa593f22fa82ff273deed20e4ef48fc95ab (HEAD -> mabranche)
Author: thouraya.louati@gmail.com <thouraya.louati@esprit.tn>
Date:   Sat Oct 2 18:35:30 2021 +0100

    Modification 3 du fichier file.txt

commit e3e870e014a32805bbac1c217f6c528618ecffdb
Author: thouraya.louati@gmail.com <thouraya.louati@esprit.tn>
Date:   Sat Oct 2 17:12:36 2021 +0100

    Modification du fichier file.txt

commit 8abc0e268b4f938142d178c10401633e28737b64
Author: thouraya.louati@gmail.com <thouraya.louati@esprit.tn>
Date:   Sat Oct 2 16:50:48 2021 +0100

    initialisation du projet
```

Démo

- Récupérer ce travail sur la branche principale : basculer sur la branche qui va recevoir (master dans notre cas), faire un merge et supprimer la branche qui vient d'être mergée (optionnel) :

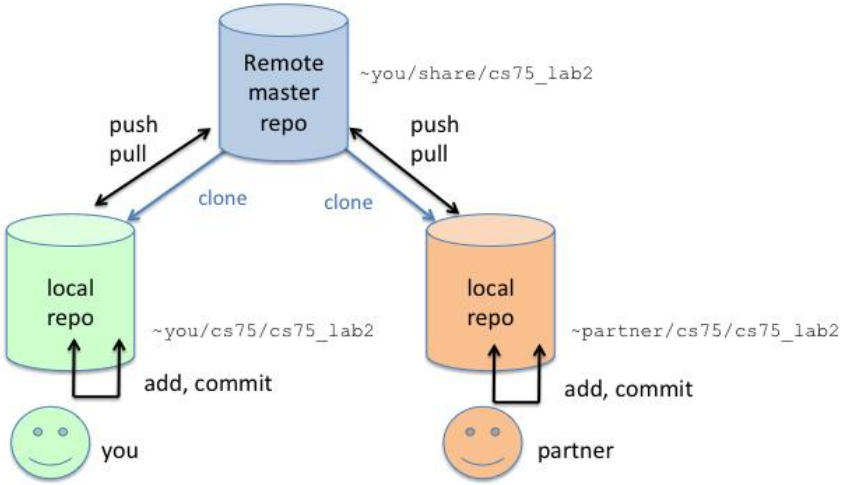
```
ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demodevops (mabranche)
$ git checkout master
Switched to branch 'master'

ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demodevops (master)
$ git branch
  mabranche
* master

ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demodevops (master)
$ git merge mabranche
```

Démo


```
ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demodevops (master|MERGING)
$ git branch -d mabranche
Deleted branch mabranche (was 136f0a4).


ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demodevops (master|MERGING)
$ git branch
* master
```

Démo

- Dépôt distant GitHub

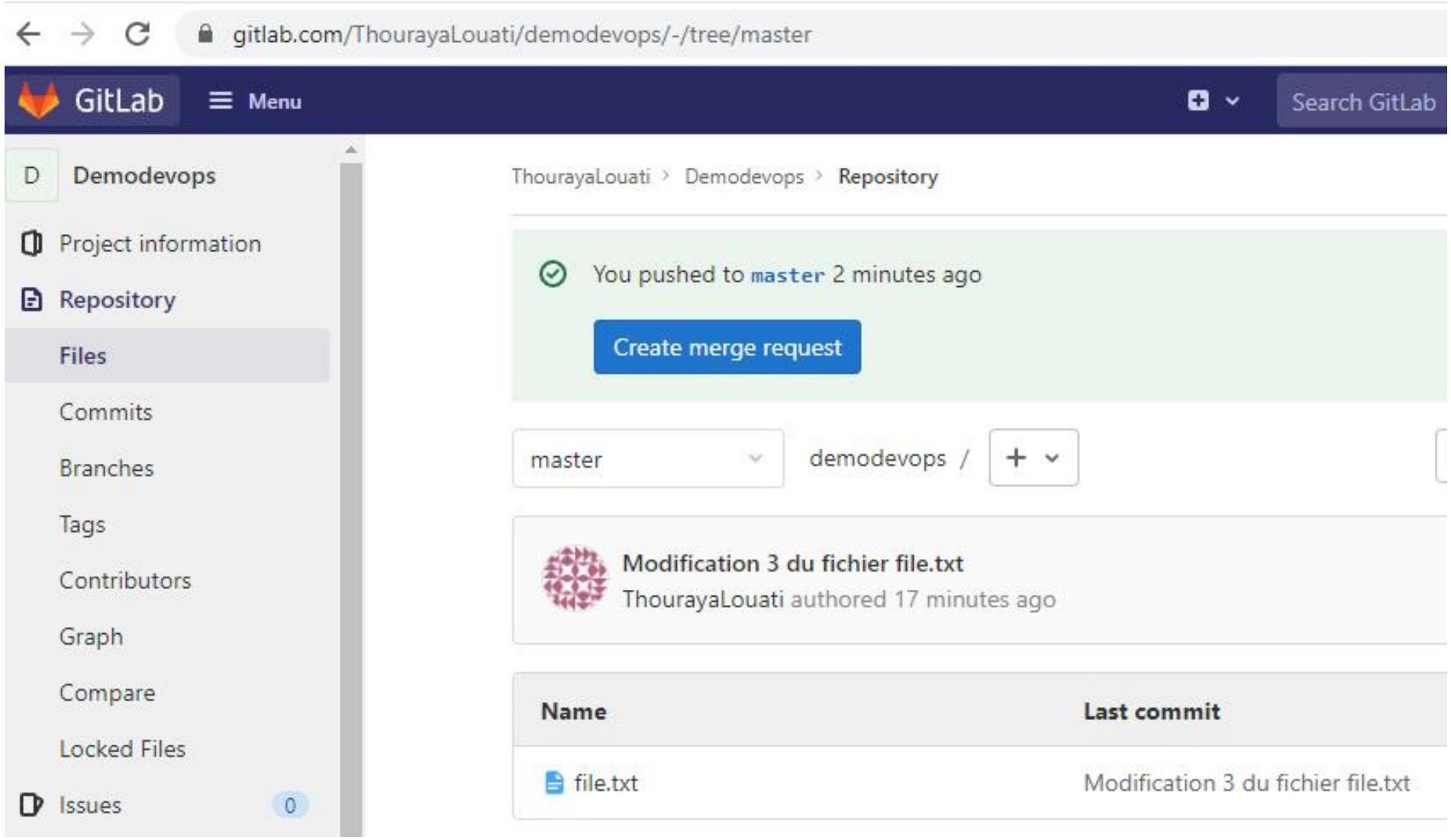


- Créer un compte sur GitHub ou GitLab
- Créer un Repository (**un seul parmi l'équipe**)
- Associer le dépôt distant à notre dépôt local (nom du dépôt distant origin)
- Enfin déposer votre projet sur le dépôt distant :

Démo


```
ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demodevops (master|MERGING)
$ git remote add origin https://gitlab.com/ThourayaLouati/demodevops.git

ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demodevops (master|MERGING)
$ git remote
origin

ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demodevops (master|MERGING)
$ git push origin master
Enumerating objects: 21, done.
Counting objects: 100% (21/21), done.
Delta compression using up to 4 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (21/21), 1.58 KiB | 231.00 KiB/s, done.
Total 21 (delta 2), reused 0 (delta 0)
remote:
remote: To create a merge request for master, visit:
remote:   https://gitlab.com/ThourayaLouati/demodevops/-
/merge_requests/new?merge_request%5Bsource_branch%5D=master
remote:
To https://gitlab.com/ThourayaLouati/demodevops.git
 * [new branch]      master -> master
```

Démo


gitlab.com/ThourayaLouati/demodevops/-/tree/master
GitLab
Menu
Search GitLab
D
Demodevops
Project information
Repository
Files
Commits
Branches
Tags
Contributors
Graph
Compare
Locked Files
Issues
0
ThourayaLouati > Demodevops > Repository
You pushed to master 2 minutes ago
Create merge request
master
demodevops /
Modification 3 du fichier file.txt
ThourayaLouati authored 17 minutes ago
Name
Last commit
file.txt
Modification 3 du fichier file.txt

Démo

- les autres membres d'un même groupe, récupèrent le code de GitHub/GitLab avec la commande ci-dessous :

```
ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demo2devops
$ git clone  https://gitlab.com/ThourayaLouati/demodevops.git
Cloning into 'demodevops'...
remote: Enumerating objects: 24, done.
remote: Counting objects: 100% (24/24), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 24 (delta 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (24/24), done.
```

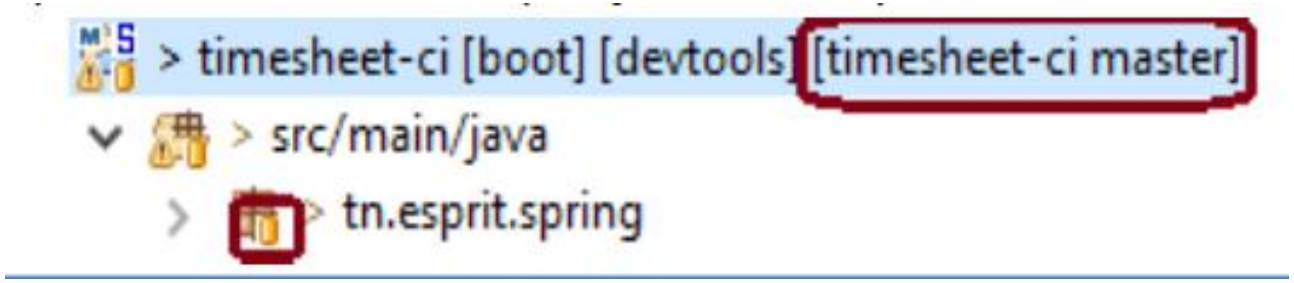
Démo

- Git clone est l'équivalent à:

```
ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demo3devops
$ git init
Initialized empty Git repository in C:/Users/ASUS/Desktop/demo3devops/.git/

ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demo3devops (master)
$ git remote add origin https://gitlab.com/ThourayaLouati/demodevops.git

ASUS@DESKTOP-41UCFB9 MINGW64 ~/Desktop/demo3devops (master)
$ git pull origin master
remote: Enumerating objects: 21, done.
remote: Counting objects: 100% (21/21), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 21 (delta 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (21/21), done.
From https://gitlab.com/ThourayaLouati/demodevops
 * branch            master     -> FETCH_HEAD
 * [new branch]      master     -> origin/master
```

Démo

- **TP Timesheet (Maven + GIT + JUnit + Log4J)**
  - Utiliser JUnit et ajouter Log4J pour tracer ce qui se passe.
- Suivez les étapes décrites ci-dessus, dans tout le support de cours Git, pour mettre le projet sur Git.
- Maintenant vous pouvez utilisez Git en ligne de commande ou sur STS (Fermez puis ouvrez votre projet sur STS).



TP1: Les commandes GIT basiques

- Tester ces commandes sur un projet et mettre les captures d'écran sur classroom.
  - init
  - clone
  - fetch
  - pull
  - commit
  - push
  - merge
  - branches

TP2: Les commandes GIT avancées

- Tester ces commandes sur un projet et mettre les captures d'écran sur classroom.
  - diff
  - tag
  - reset
  - revert
  - git-flow

Références

- https://git-scm.com/book/fr/v2/D%C3%A9marrage-rapide-%C3%80-propos-de-la-gestion-de-version
- https://doc.ubuntu-fr.org/git
- https://git-scm.com/docs/git-clone/fr
- Cours GIT, Mourad HASSINI, ESPRIT.
- https://enacit.epfl.ch/cours/git/

GIT

2021-2022

ESPRIT thouraya.louati@esprit.tn UP ASI (Bureau E204)