

*Toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l'auteur ou de ses ayant droit ou ayant cause est illicite. Il en est de même pour la traduction, l'adaptation ou la transformation, l'arrangement ou la reproduction par un art ou un procédé quelconque,*



# Versioning Cours

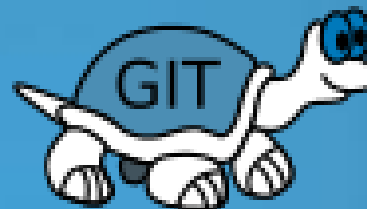
Naby Daouda Diakite



GitLab



GitHub



# Présentation des participants

- Formateur
- Stagiaires
- Déroulement de la formation  
(*combinaison des parties  
théoriques et pratiques*)
- Echange sur des aspects  
transverses de nos métiers
- Partage des expériences



# Programme

I. Introduction

II. Outils de versioning

III. Git

IV. TortoiseGit

# I. Introduction

- I.I. Définition
- I.II. Avant les outils de versioning
- I.III. Serveurs de versioning

# I. Introduction

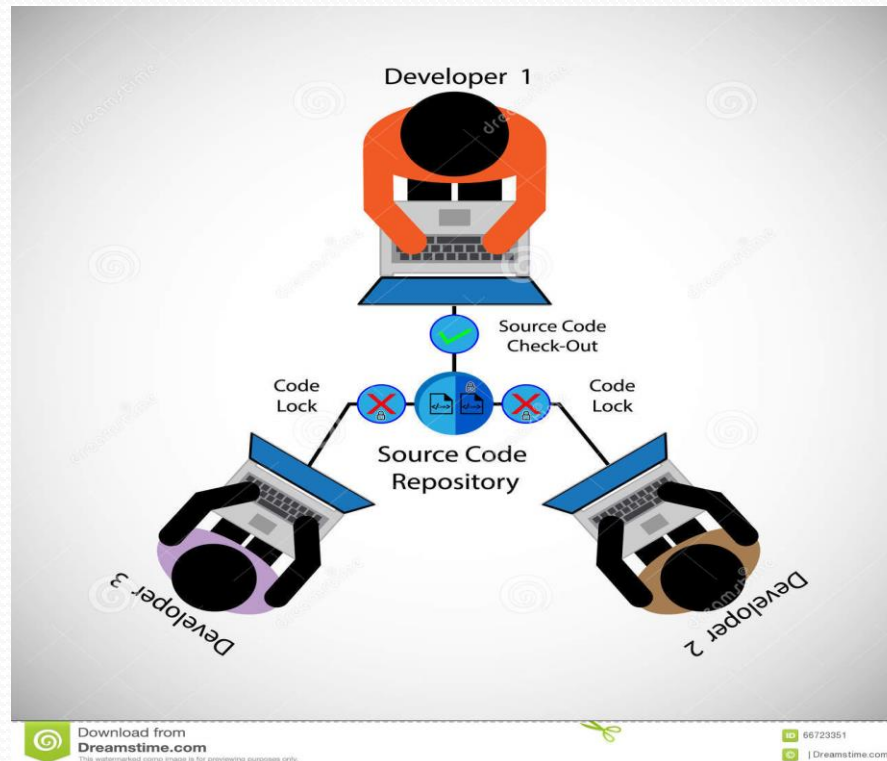
## I.I. Définition

- Un logiciel de gestion de versions (ou VCS en anglais, pour *Version Control System*) est un logiciel qui permet de stocker un ensemble de fichiers en conservant la chronologie de toutes les modifications qui ont été effectuées dessus.
- Il permet notamment de retrouver les différentes versions d'un lot de fichiers connexes.

# I. Introduction

## I.1. Définition

- Il facilite le travail en équipe, notamment durant les développements.



# I. Introduction

## I.II. Avant les outils de versioning (1)

- C'était assez compliqué de construire de gros projets en équipe.
- Les techniques de travail étaient variées :
  - Partage sur clé USB
  - Envoi de fichiers par mail
  - Etc ..

# I. Introduction

## I.II. Avant les outils de versioning (2)

- Ca finissait dès fois très mal !!



T'as perdu quoi ? ..



# I. Introduction

## I.III. Serveurs de versioning (1)

- Ce sont des outils qui permettent de mettre en place un système de versioning au sein de l'équipe.
- Ils ne sont pas indispensables
- Mais ils fournissent une interface graphique et des fonctionnalités qui permet de gérer plus facilement le versioning.
  - Création de compte utilisateur
  - Création de groupe
  - Accès en ligne aux fichiers et aux historiques
  - Etc ..

# I. Introduction

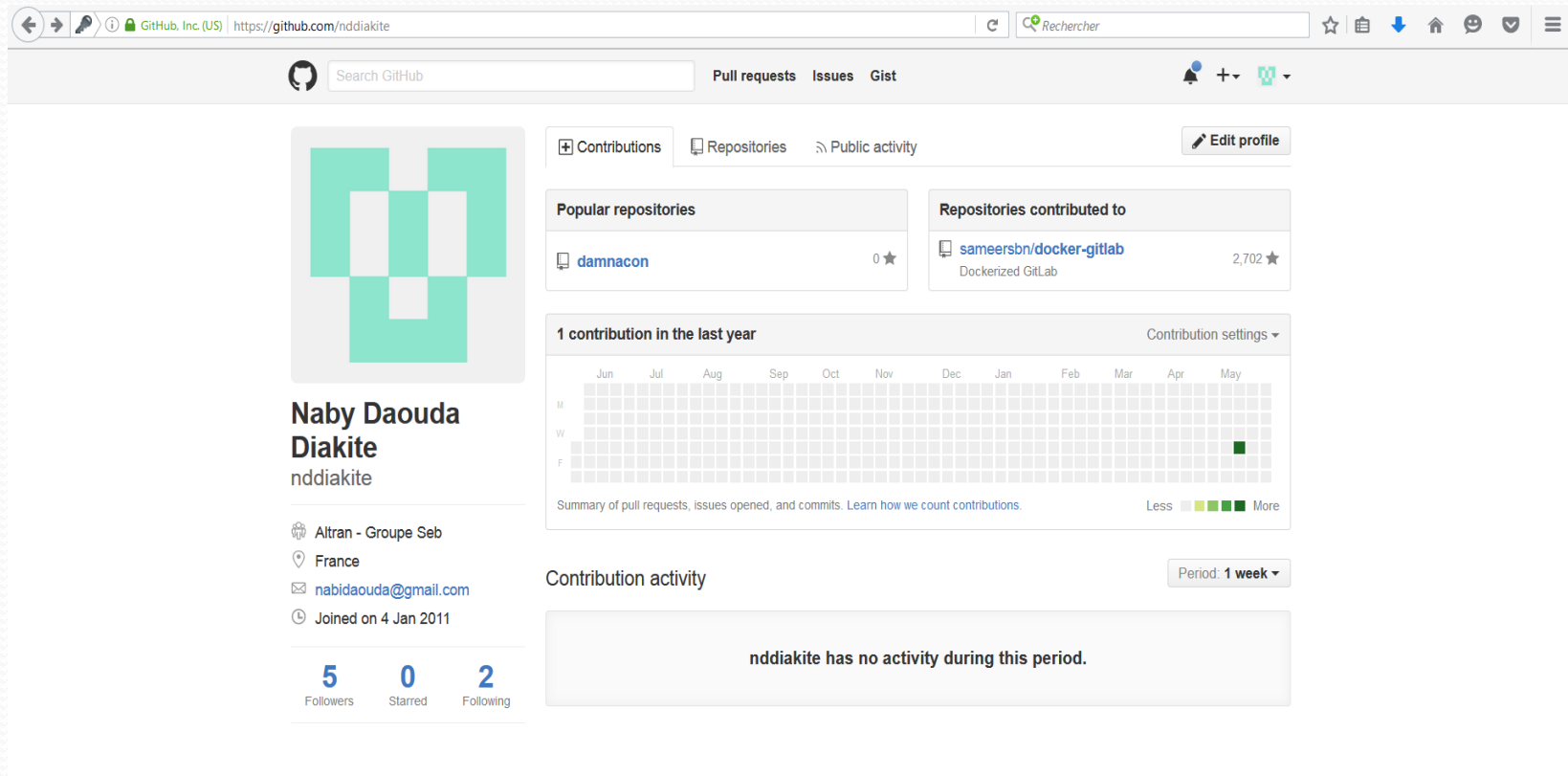
## I.III. Serveurs de versioning (2)

- Ils sont nombreux :
  - Système de versioning en ligne
    - Github
    - Bitbucket
  - Système de versioning sur son propre serveur
    - Gitlab
    - Gitweb
    - Gitolite

# I. Introduction

## I.III. Serveurs de versioning (3)

- Interface de Github



The screenshot displays the GitHub profile of Naby Daouda Diakite (nndiakite). The profile includes a bio, location (France), email (nabidaouda@gmail.com), and join date (4 Jan 2011). The statistics section shows 5 Followers, 0 Starred, and 2 Following. The main content area features a 'Popular repositories' section with 'damnacon' (0 stars), a 'Repositories contributed to' section with 'sameersbn/docker-gitlab' (2,702 stars), and a 'Contribution activity' section showing no activity during the selected period (1 week).



# I. Introduction

## I.III. Serveurs de versioning (4)

- Interface de Gitlab

The screenshot displays the GitLab Admin Area interface. On the left is a dark sidebar with navigation links: Overview, Projects, Users, Groups, Deploy Keys, Runners (0), Builds (0), Logs, Messages, Hooks, Background Jobs, Applications, Service Templates, Labels, Abuse Reports (0), and a user profile for 'nddiakite'. The main content area is titled 'Admin Area' and includes a search bar and utility icons. It is divided into three columns: Statistics, Features, and Components. The Components column has a red 'update asap' button. Below these columns are three large cards for Projects (27), Users (16), and Groups (4), each with a 'NEW PROJECT', 'NEW USER', or 'NEW GROUP' button. At the bottom, there are links for 'Latest projects', 'Latest users', and 'Latest groups'.

Statistics		Features		Components	
Forks	0	Sign up	●	GitLab	8.3.2
Issues	0	LDAP	⏻	GitLab Shell	2.6.9
Merge Requests	0	Gravatar	●	GitLab API	v3
Notes	0	OmniAuth	⏻	Ruby	2.1.8p440
Snippets	0	Reply by email	⏻	Rails	4.2.4
SSH Keys	1				
Milestones	0				
Active Users	14				

Category	Count	Action
Projects	27	NEW PROJECT
Users	16	NEW USER
Groups	4	NEW GROUP

## II. Outils de versioning

- II.I. Types d'outil
- II.II. CVS
- II.III. SVN
- II.IV. Git

# II. Outils de versioning

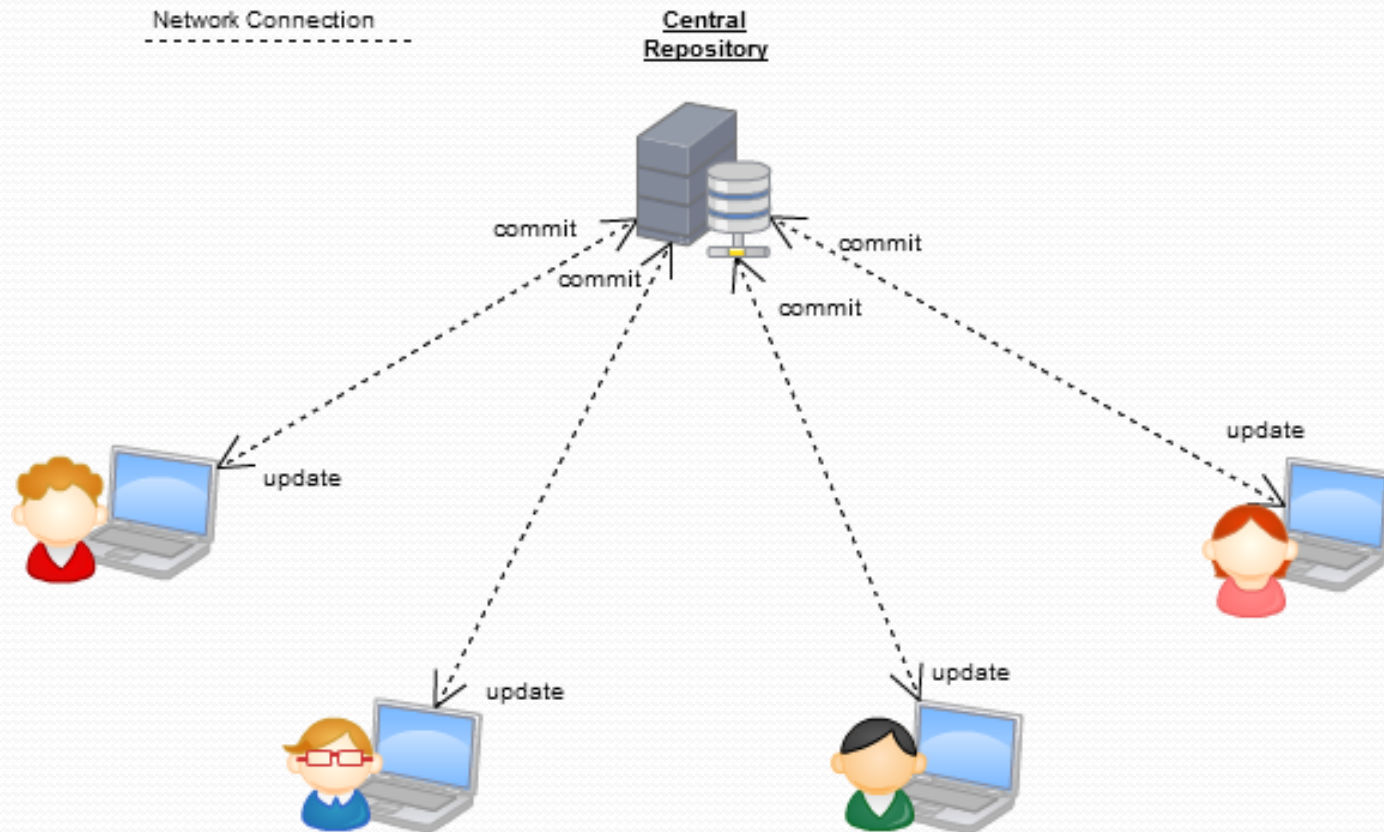
## II.1. Types (1)

- Les types sont :
  - Les outils de versioning « centralisés » : les premiers outils étaient sur ce modèle :
    - CVS
    - SVN
    - Etc..
  - Les outils de versioning « décentralisés » : les nouveaux outils sont sur ce modèle :
    - Git
    - Mercurial
    - Etc..

# II. Outils de versioning

## II.1. Types (2)

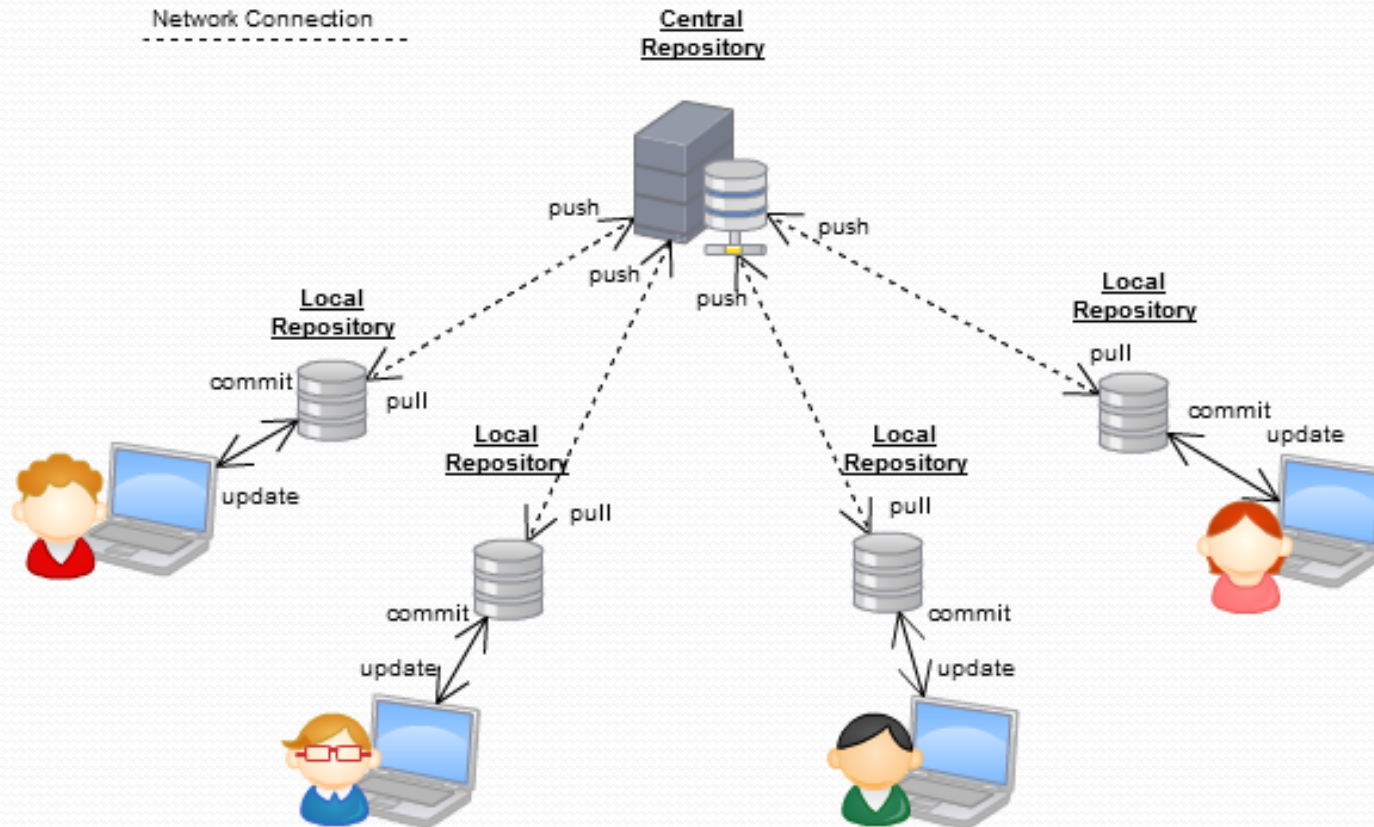
- Les outils de versioning « centralisés »



# II. Outils de versioning

## II.1. Types (3)

- Les outils de versioning « décentralisés »





# II. Outils de versioning

## II.II. CVS

- CVS : Versions System (modèle centralisé)
- Outil de versioning créé en 1990
- Beaucoup utilisé par les projets de logiciels libres.

# II. Outils de versioning

## II.III. SVN

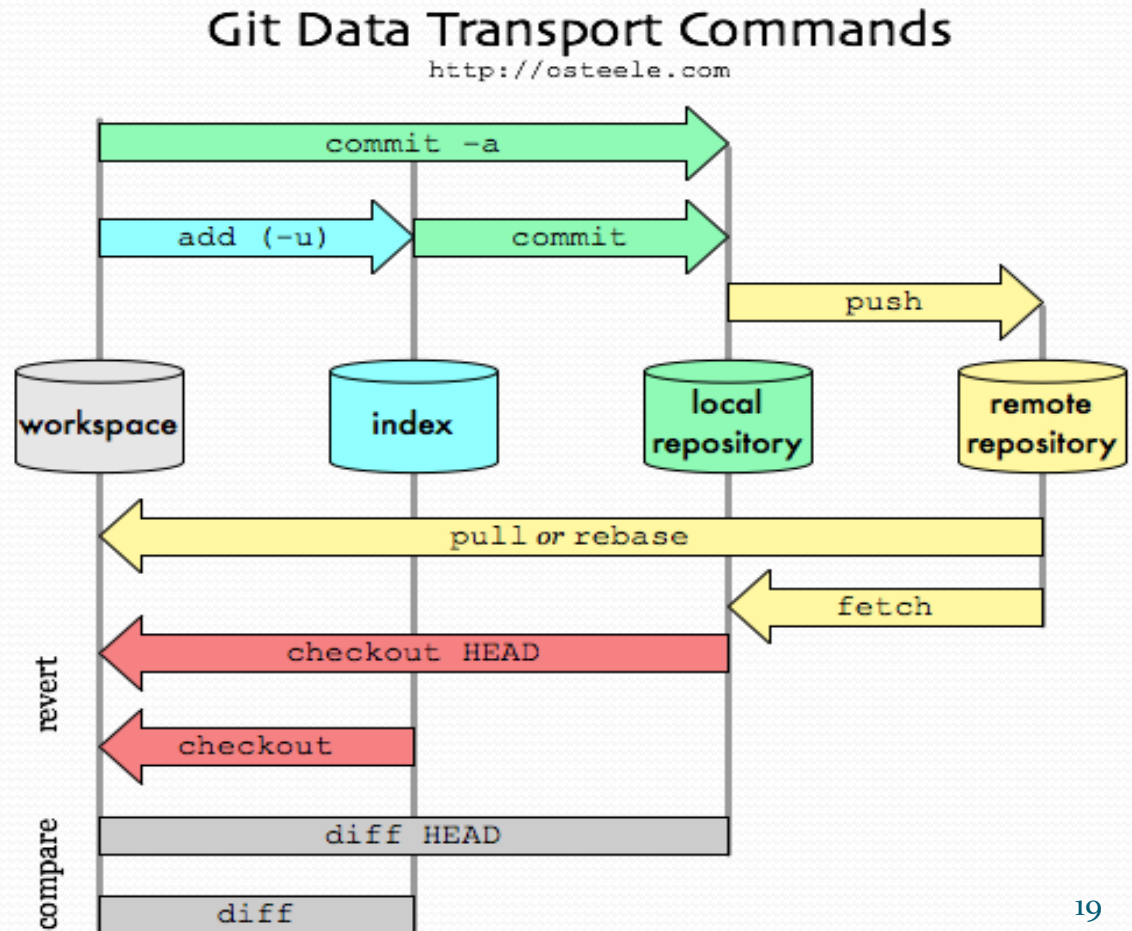
- Subversion en abrégé SVN (*modèle centralisé*)
- Outil de versioning, distribué sous licence Apache et BSD. Il a été conçu pour remplacer CVS.
- Apports
  - Le concept de « commit »
  - Le renommage de fichier
  - Le déplacement de fichier
  - La conservation des métadonnées (droits sur fichiers..)
  - Etc ..

# II. Outils de versioning

## II.IV. Git (1)

- Git (*modèle décentralisé*)

- Fonctionnement



# II. Outils de versioning

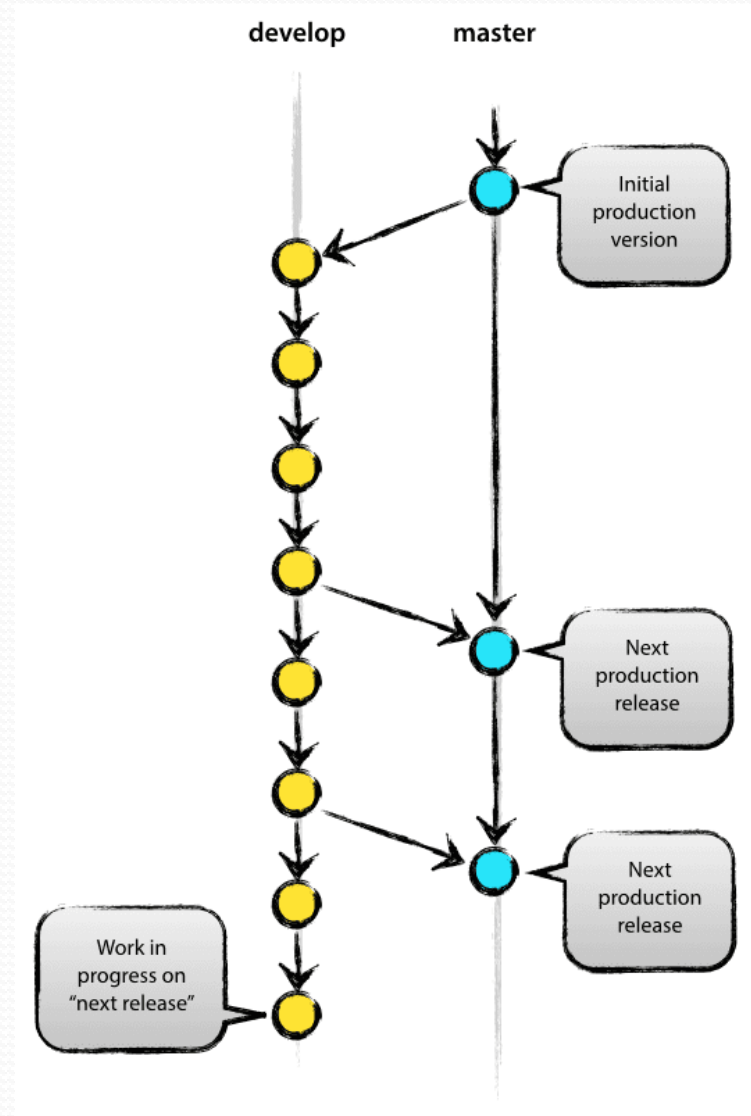
## II.IV. Git (2)

- Concept de « branche »
  - Les branches permettent de gérer plusieurs versions du projet en parallèle et dans le temps
  - La branche par défaut est « *master* »

# II. Outils de versioning

## II.IV. Git (3)

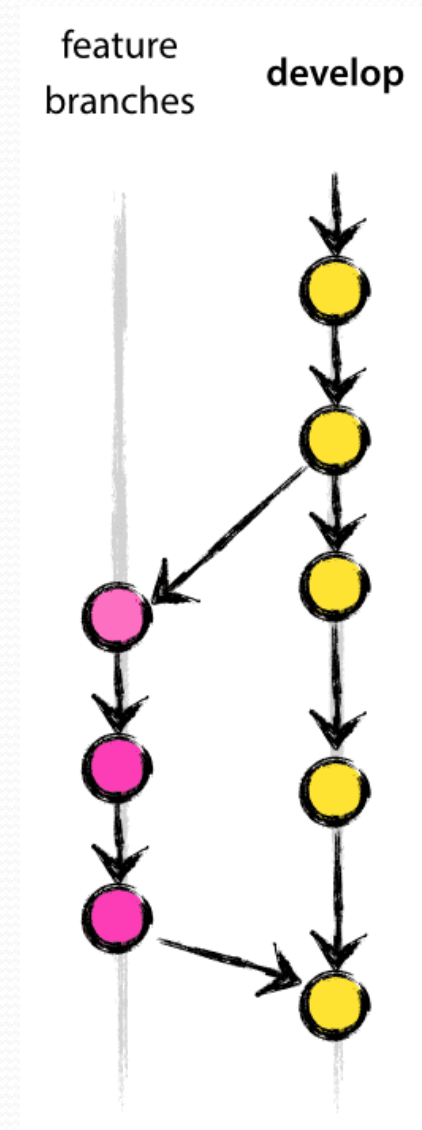
- Branching model
  - Branche « master »
  - Branche « develop »



# II. Outils de versioning

## II.IV. Git (4)

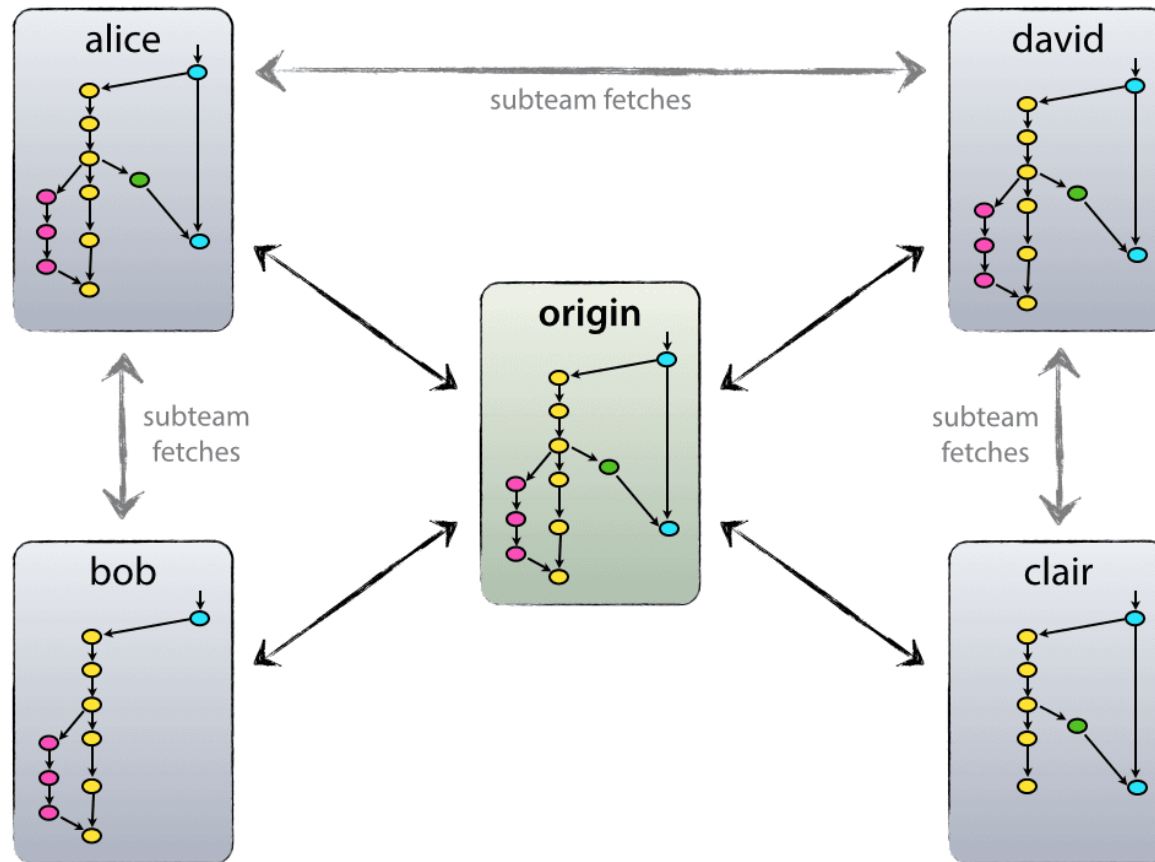
- Branching model
  - **Branches « feature »**
    - Exemple : « feature/authenticate »



# II. Outils de versioning

## II.IV. Git (5)

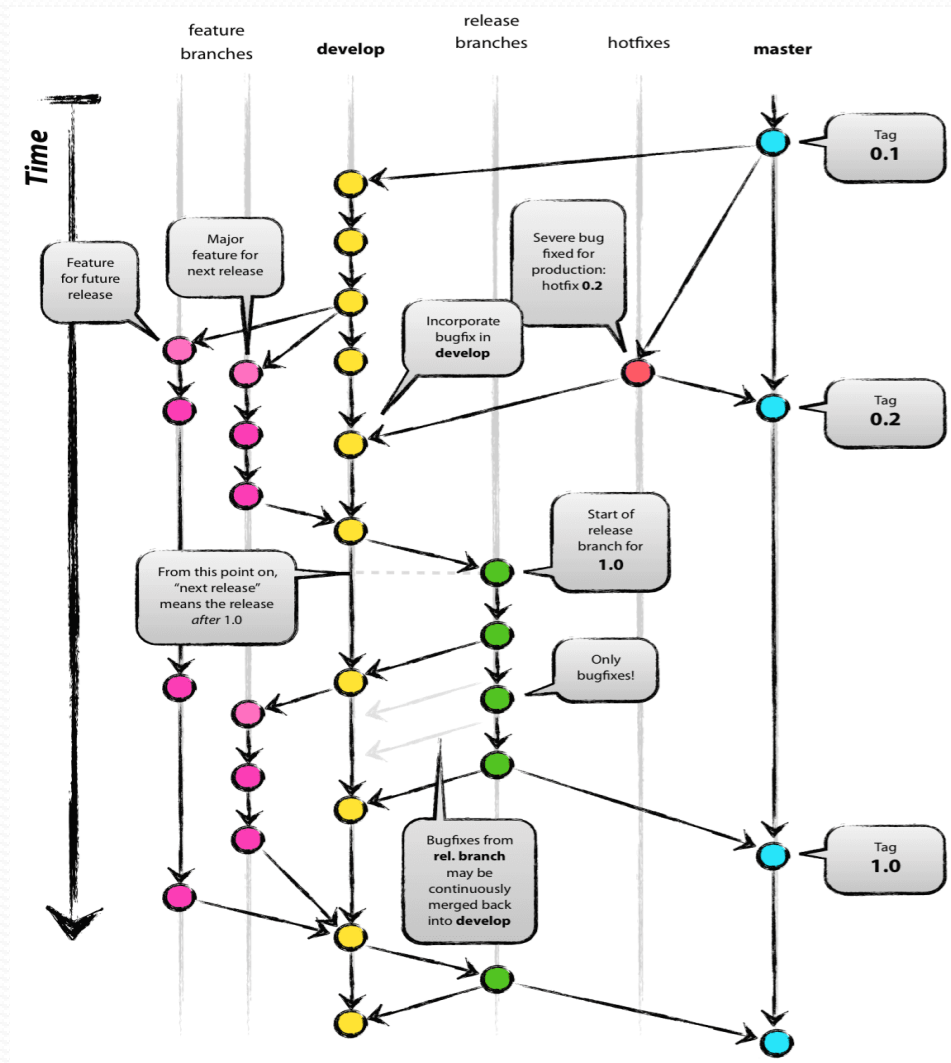
- Branching model ➔ **Diversité sur les postes des développeurs**



# II. Outils de versioning

## II.IV. Git (6)

- Branching model





# III. Git

- III.I. Installation
- III.II. Commandes de base
- III.III. Création d'un repo distant sur le serveur
- III.IV. Récupération d'un repo en local
- III.V. Ajout d'un nouveau fichier
- III.VII. Publication des modifications
- III.VI. Mise à jour de la version locale
- III.VIII. Historique des modifications
- III.IX. Gestion des conflits
- III.X. Création d'une branche
- III.XI. Merge d'une branche
- III.XII. Suppression d'une branche

# III. Git

## III.1. Installation

- Allez sur la page officielle de Git : <https://git-scm.com/downloads>
- Sélectionnez la version (fichier .exe) correspondant à votre système d'exploitation
- Lancez cet exécutable et suivez les différentes étapes



# III. Git

## III.II. Commandes de base (1)

- Commande « init » ==> transformer un répertoire standard en repo git
  - Initiation d'un repo
- Commande « clone »
  - Création d'une copie locale d'un repo distant
- Commande « add »
  - Ajout de fichier(s)
- Commande « commit »
  - Publication des modifications sur le repo local

# III. Git

## III.II. Commandes de base (2)

- Commande « pull »
  - Mise à jour du repo local depuis le repo distant
- Commande « push »
  - Publication des modifications sur le repo local
- Commande « status »
  - Information sur les modifications en cours (copie de travail)
- Commande « diff »
  - Information sur les conflits en cours
- Commande « log »
  - Historique des commits

# III. Git

## III.II. Commandes de base (3)

- Commande « branch »
  - Création d'une branche
- Commande « checkout »
  - Se positionner sur une branche
  - Création d'une branche avec l'option « checkout -b »

# III. Git

## III.II. Commandes de base (4)

- Liens utiles
  - <http://rogerdudler.github.io/git-guide/index.fr.html>
  - <https://confluence.atlassian.com/bitbucketserver044/git-resources/basic-git-commands>

# III. Git

## III.III. Création d'un repo distant sur le serveur (1)

- Il faut se connecter au serveur
- Créer un répertoire pour le repo et se placer dans ce répertoire
- Exécuter la commande : `git init`
- Fournir l'url vers ce repo

# III. Git

## III.III. Création d'un repo distant sur le serveur (2)

- Outil de gestion de versioning comme Gitlab

New Project

Project path   /

Want to house several dependent projects under the same namespace? [Create a group](#)

Import project from

Description (optional)

Visibility Level (?) ☒ ☐ ☐

☒ Private  
Project access must be granted explicitly to each user.

☐ Internal  
The project can be cloned by any logged in user.

☐ Public  
The project can be cloned without any authentication.



# III. Git

## III.IV. Récupération d'un repo en local

- Il faut se placer dans un répertoire quelconque
- Exécuter la commande « clone »

```
git clone repo_url
```

### **Exemple :**

```
git clone joe@192.98.63.22:/var/lib/my_repo
```

```
git clone https://gitlab.ovni.fr/formation-java/git-tp-n1.git
```

# III. Git

## III.V. Ajout d'un nouveau fichier

- Il faut se placer dans le répertoire du repo pour exécuter la commande « add »
- Ajouter un fichier

```
git add <filename>
```

**Exemple :**

```
git add user.java
```

- Ajouter plusieurs fichiers

```
git add *
```

# III. Git

## III.VI. Publication des modifications (1)

- Les modifications sont :
  - Les nouveaux fichiers
  - Les fichiers supprimés
- Il y a 2 niveaux de publication
  - Publication sur le repo local
    - Les autres membres du projet ne voient pas les modifications
  - Publication sur le repo distant
    - Les autres membres du projet voient les modifications

# III. Git

## III.VI. Publication des modifications (2)

- Publication sur le repo local
  - Il faut se placer dans le répertoire du repo pour exécuter la commande « commit » ==> quand on fait un commit, il faut mettre un message

```
git commit -m "Commit message"
```

- Publication sur le repo local (avec ajout des nouveaux fichiers dans l'index)
  - Il faut se placer dans le répertoire du repo pour exécuter la commande « commit » avec l'option « -a » en plus

```
git commit -a -m "Commit message"
```

# III. Git

## III.VI. Publication des modifications (3)

- Publication sur le repo distant
  - Il faut se placer dans le répertoire du repo pour exécuter la commande « push »

```
git push remoteName branchName
```

reference de la  
branche distante

### **Exemple :**

```
git push origin master
```

# III. Git

## III.VII. Mise à jour de la version locale

- Il faut se placer dans le répertoire du repo pour exécuter la commande « pull »

```
git pull remoteName branchName
```

**Exemple :**

```
git pull origin master
```

# III. Git

## III.VIII. Historique des modifications

- Il faut se placer dans le répertoire du repo pour exécuter la commande « status »

```
git status
```

`git log` ==> permet de mettre tous les numéros de commit, les messages et les auteurs

# III. Git

## III.IX. Gestion de conflits (1)

- Il faut se placer dans le répertoire du repo pour exécuter la commande « status »

```
git diff
```

- Cela affichera la liste des fichiers en conflit
- Il faut maintenant aller dans chacun de ces fichiers et résoudre les conflits détectés.



# III. Git

## III.IX. Gestion de conflits (2)

- Enfin indiquer que les conflits ont été résolus avec la commande suivante

```
git add <filename>
```

**Exemple :**

```
git add user.java
```

# III. Git

## III.X. Création et changement de branche (1)

- Il faut se placer dans le répertoire du repo et sur la branche de référence souhaiter pour exécuter la commande « checkout »

```
git branch <branchname>
```

### **Exemple :**

```
git branch feature_user ==> équivaut à un Ctrl+V
```

```
git checkout feature_user ==> pour se déplacer
```

```
git checkout -b <branchname> ==> pour faire les deux en même temps
```

### **Exemple :**

```
git checkout -b feature_user
```

# III. Git

## III.X. Création et changement de branche (2)

- Pour publier la branche sur le repo distant
  - Il faut se placer dans le répertoire du repo pour exécuter la commande « checkout »

```
git push origin <branchname>
```

### **Exemple :**

```
git push origin feature_user
```

# III. Git

## III.XI. Merge d'une branche

- Il faut se placer dans le répertoire du repo pour exécuter la commande « merge »
  - Faire la commande « checkout » sur la branche depuis laquelle on veut merger
  - Ensuite exécuter la commande « merge »

```
git checkout <branchname>
```

```
git merge <branchNameToMerge>
```

==> pour faire le lien entre la branche qu'on vient de développer  
et celle qu'on veut relier

### **Exemple :**

```
git checkout master
```

```
git merge feature_user
```

# III. Git

## III.XII. Suppression d'une branche (1)

- Suppression d'une branche locale
  - Il faut se placer dans le répertoire du repo pour exécuter la commande « branch »

```
git branch -d <branchname>
```

### **Exemple :**

```
git branch -d feature_user
```

# III. Git

## III.XII. Suppression d'une branche (2)

- Suppression d'une branche distante
  - Il faut se placer dans le répertoire du repo pour exécuter la commande « push »

```
git push origin :<branchname>
```

### **Exemple :**

```
git push origin :feature_user
```

# IV. TortoiseGit

- IV.I. Installation
- IV.II. Récupération d'un repo en local
- IV.III. Ajout d'un nouveau fichier
- IV.IV. Publication des modifications
- IV.V. Mise à jour de la version locale

# IV. TortoiseGit

## IV.1. Installation

- Allez sur la page officielle de TortoiseGit :  
<https://tortoisegit.org/download/>
- Sélectionnez la version (fichier .exe) correspondant à votre système d'exploitation
- Lancez cet exécutable et suivez les différentes étapes



The screenshot shows the 'Download' page of the TortoiseGit website. At the top, there is a navigation bar with links for 'About', 'Download' (which is highlighted), 'Support', and 'Contribute'. Below the navigation bar, the page title is 'TortoiseGit.org » Download'. The main heading is 'Download'. The text states 'The current stable version is: 2.1.0.0'. It provides a link to 'release notes' for detailed information. A note mentions that this version doesn't run on Windows XP and Server 2003, suggesting version 1.8.16.0 instead. There is a 'Donate' link. A warning states: 'Please make sure that you choose the right installer for your PC, otherwise the setup will fail.' Below this, there are two download links: 'Download TortoiseGit 2.1.0.0 - 32-bit (~16 MB)' and 'Download TortoiseGit 2.1.0.0 - 64-bit (~19 MB)'. A 'Known issue' section mentions that TortoiseGit may crash on startup if the git config is broken or libgit emits warnings/errors, with a link to issue #2745 and a note that a hotfix is available. Finally, it advises checking the latest preview release and a link to 'What to do if a crash happened?' before reporting an issue.

TortoiseGit  
Windows Shell Interface to Git

About Download Support ▾ Contribute

TortoiseGit.org » Download

## Download

The current stable version is: 2.1.0.0

For detailed info on what's new, read the [release notes](#).

FAQ: [System prerequisites and installation](#) - This version doesn't run on Windows XP and Server 2003, use [1.8.16.0](#) instead.

[Donate](#)

Please make sure that you choose the right installer for your PC, otherwise the setup will fail.

for 32-bit OS	for 64-bit OS
<a href="#">Download TortoiseGit 2.1.0.0 - 32-bit (~16 MB)</a>	<a href="#">Download TortoiseGit 2.1.0.0 - 64-bit (~19 MB)</a>

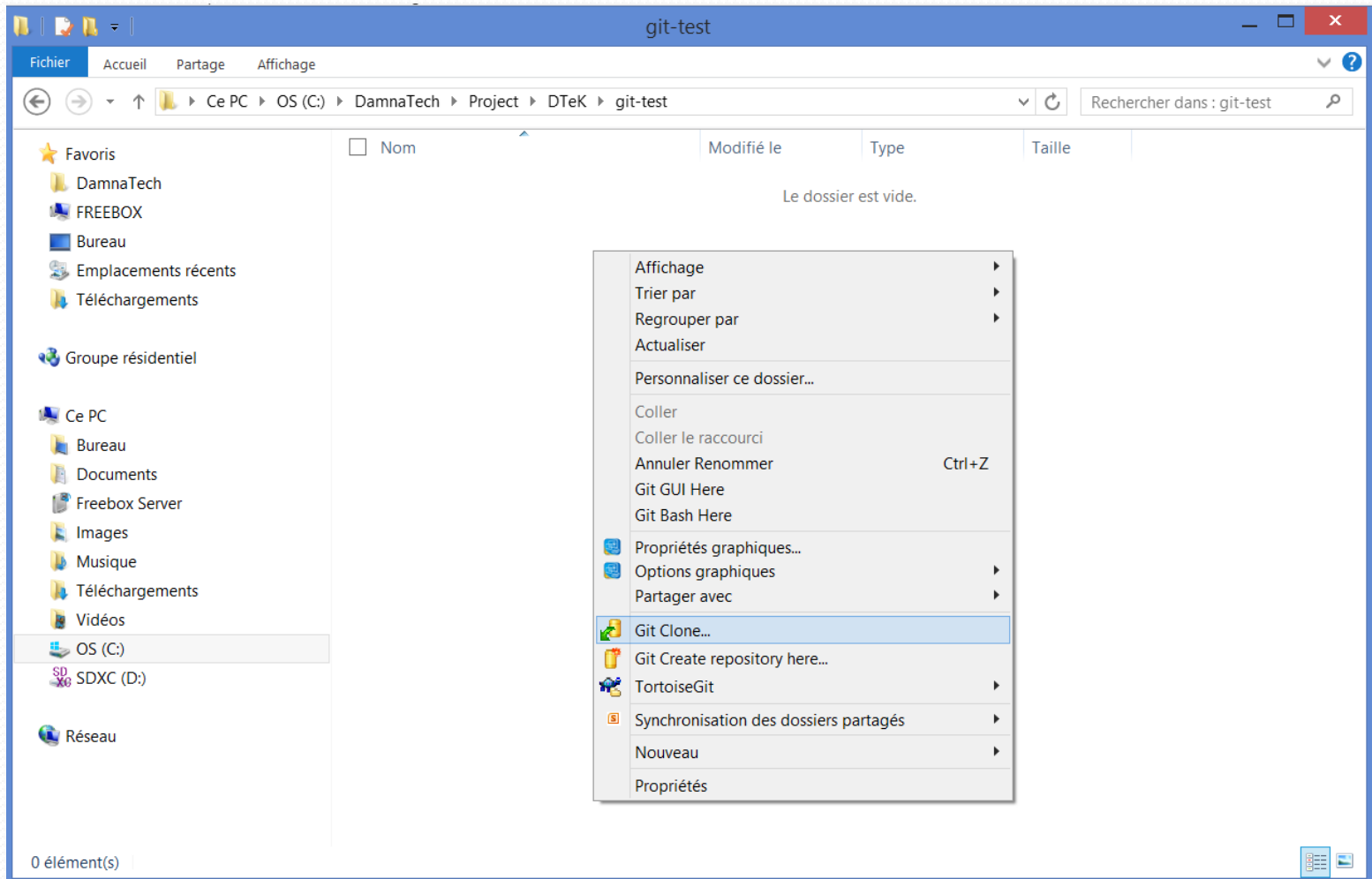
**Known issue:** TortoiseGit may crash on startup if your git config is broken or libgit emits warnings/errors (see [issue #2745](#)). [Hotfix is available](#).

Before reporting an issue, please check that your problem isn't fixed in our latest [preview release](#). Also see [What to do if a crash happened?](#)



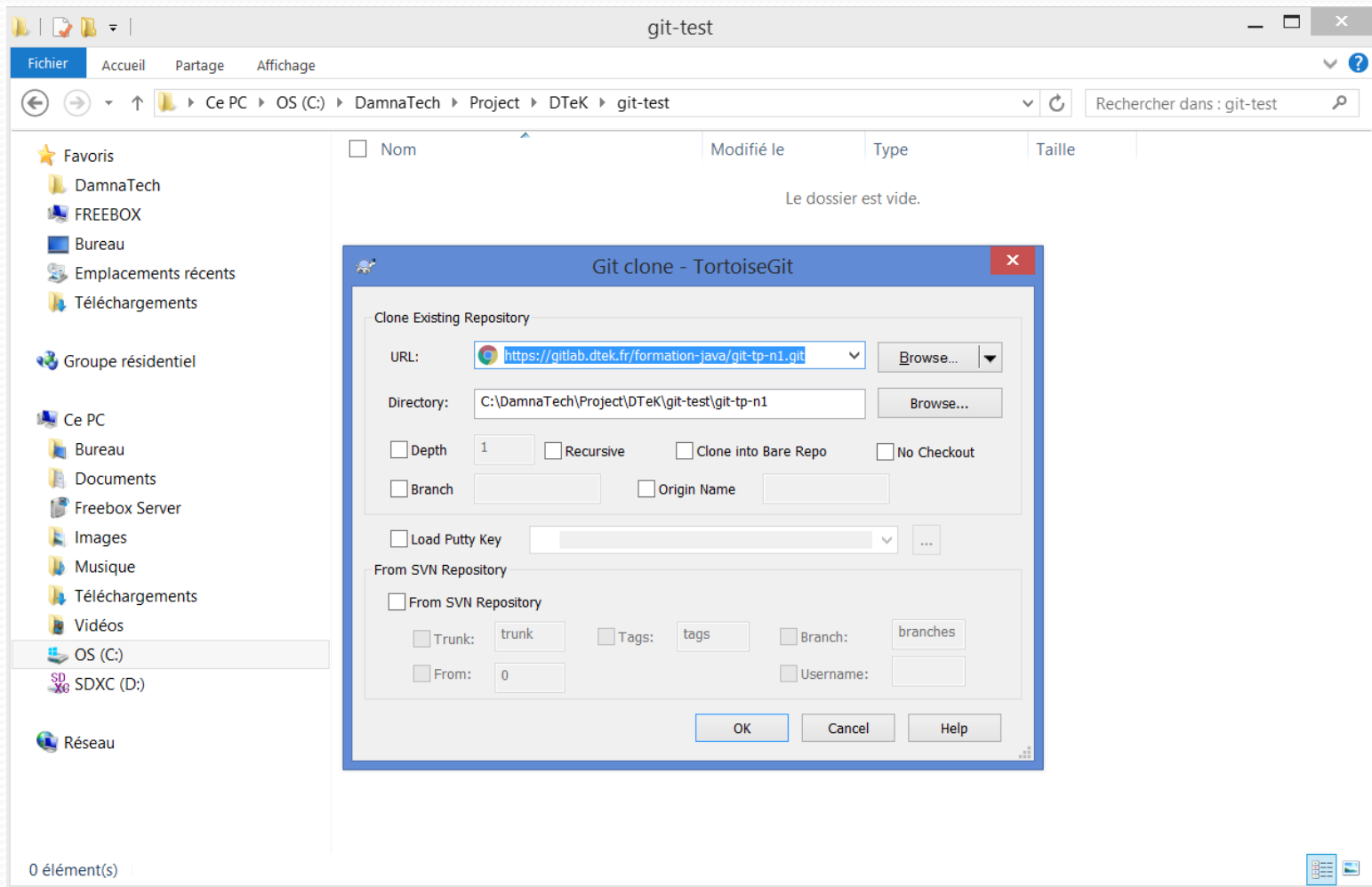
# IV. TortoiseGit

## IV.II. Récupération d'un repo en local (1)



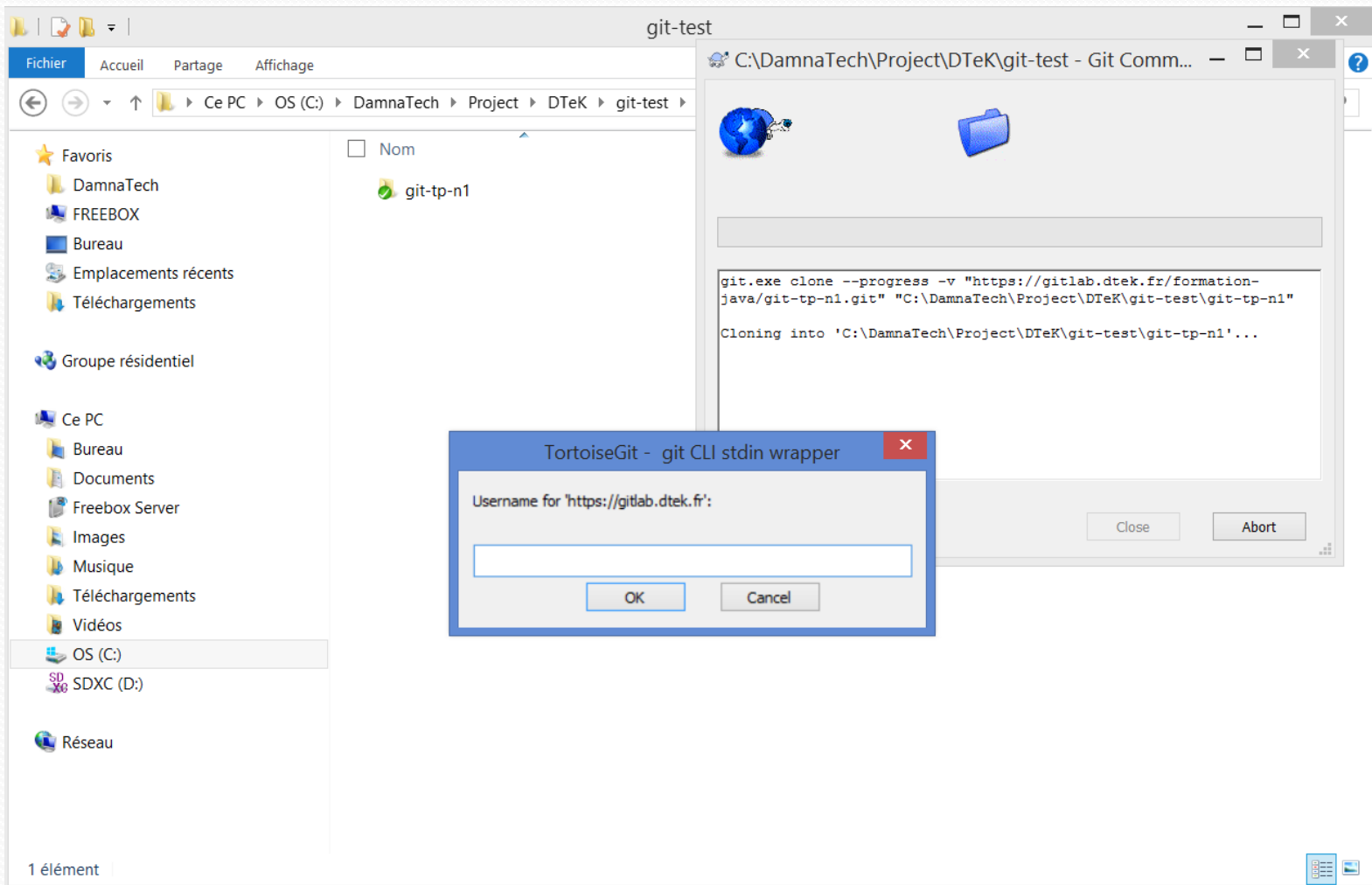
# IV. TortoiseGit

## IV.II. Récupération d'un repo en local (2)



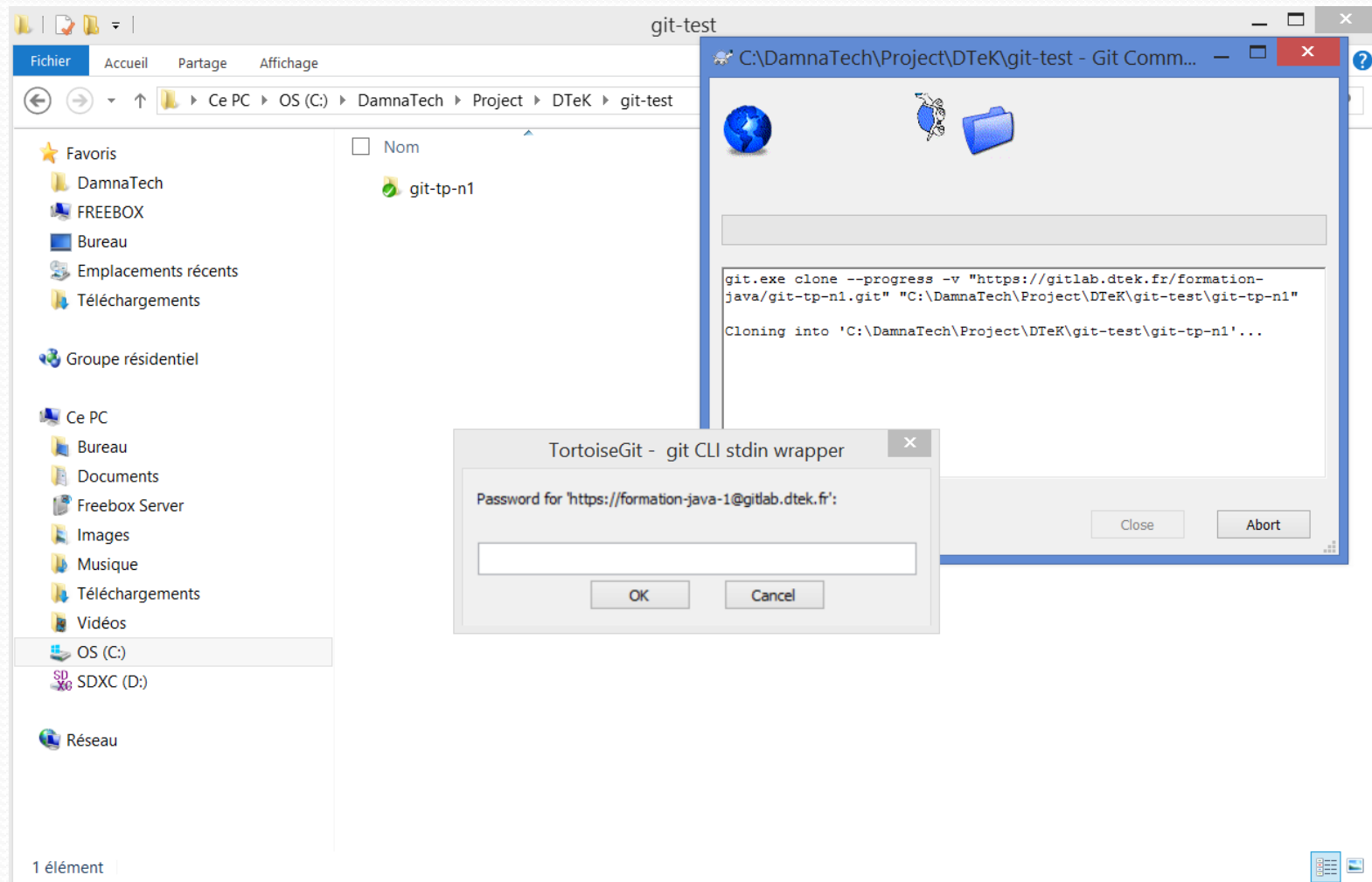
# IV. TortoiseGit

## IV.II. Récupération d'un repo en local (3)



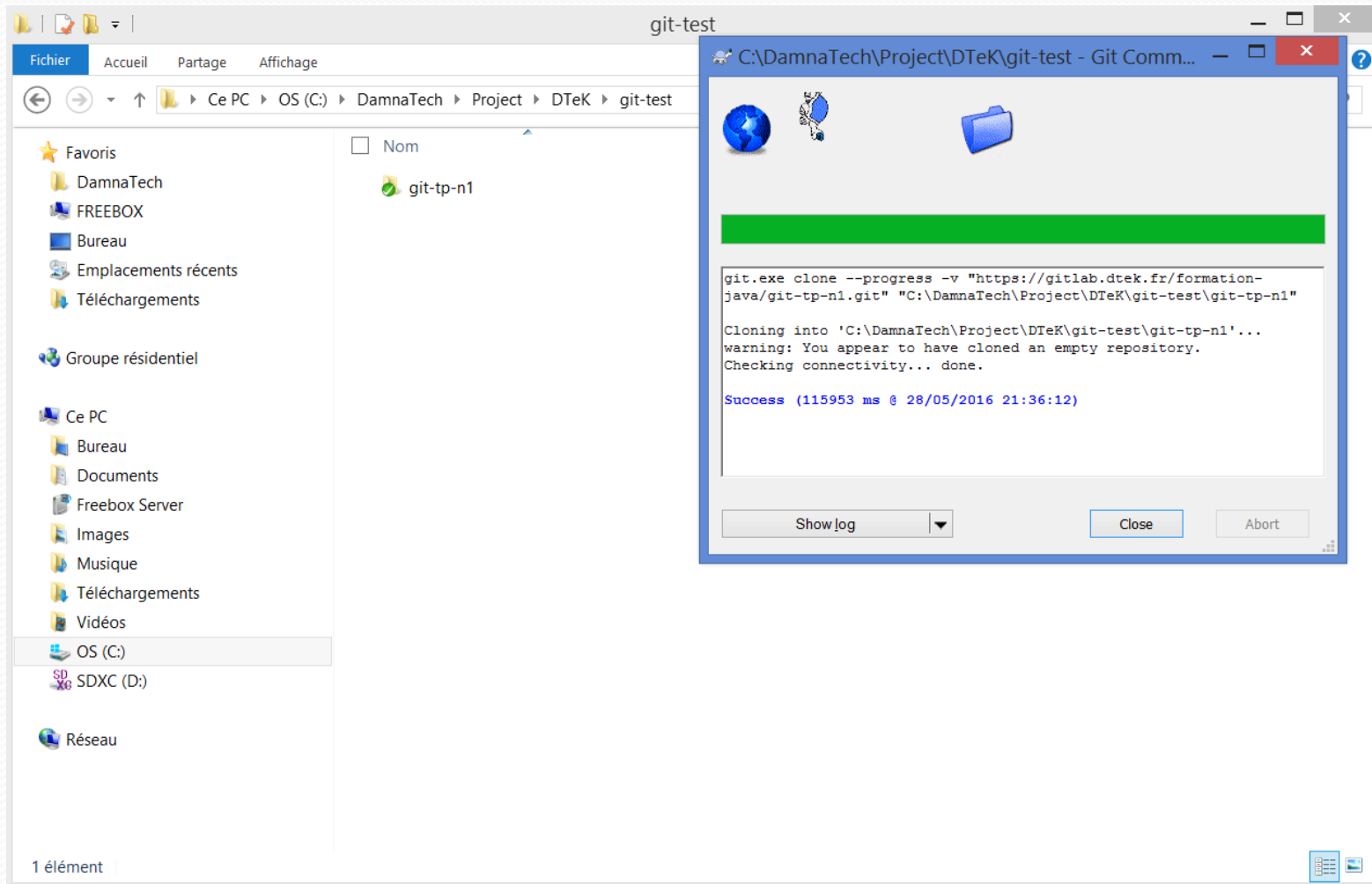
# IV. TortoiseGit

## IV.II. Récupération d'un repo en local (4)



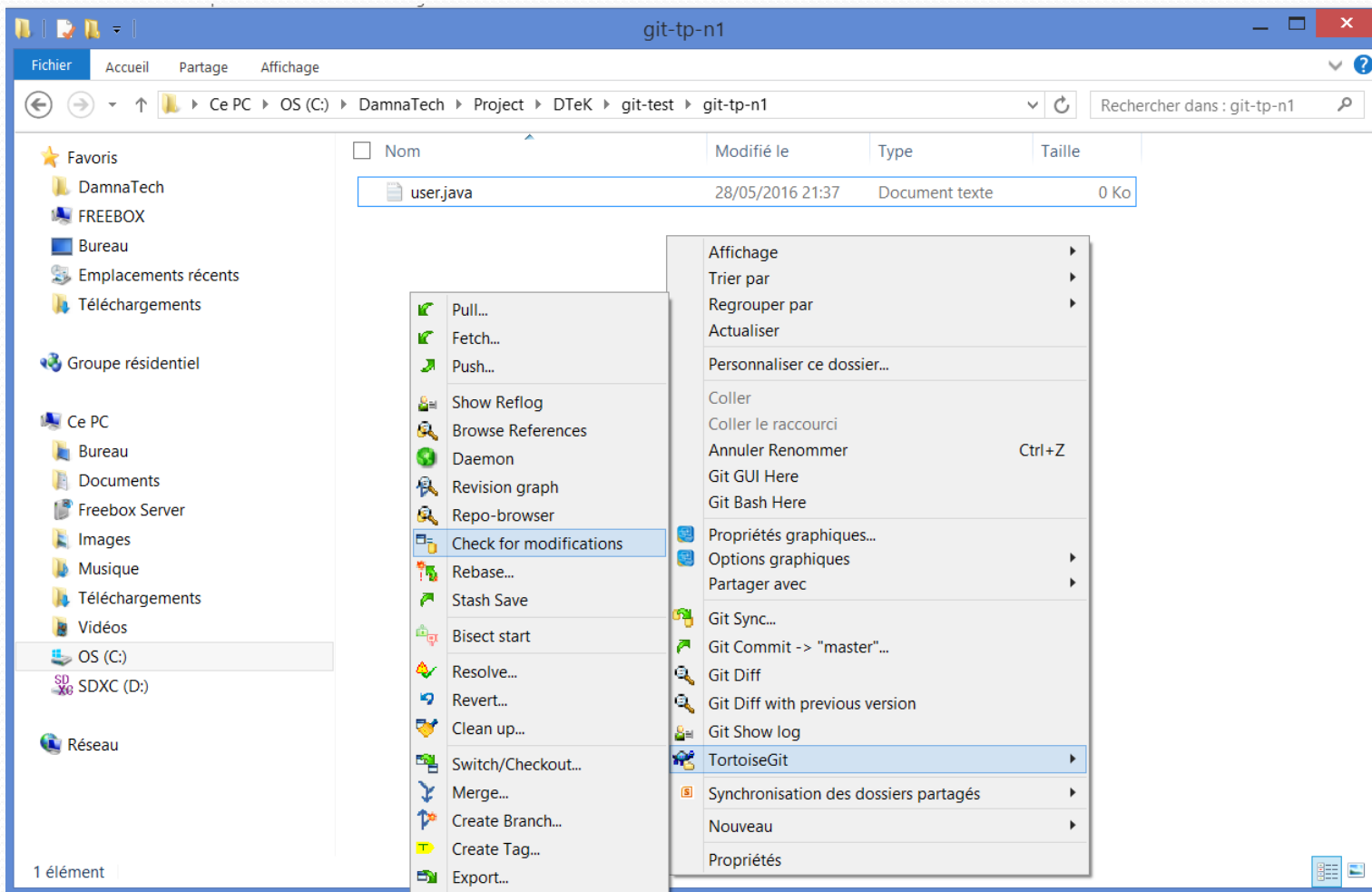
# IV. TortoiseGit

## IV.II. Récupération d'un repo en local (5)



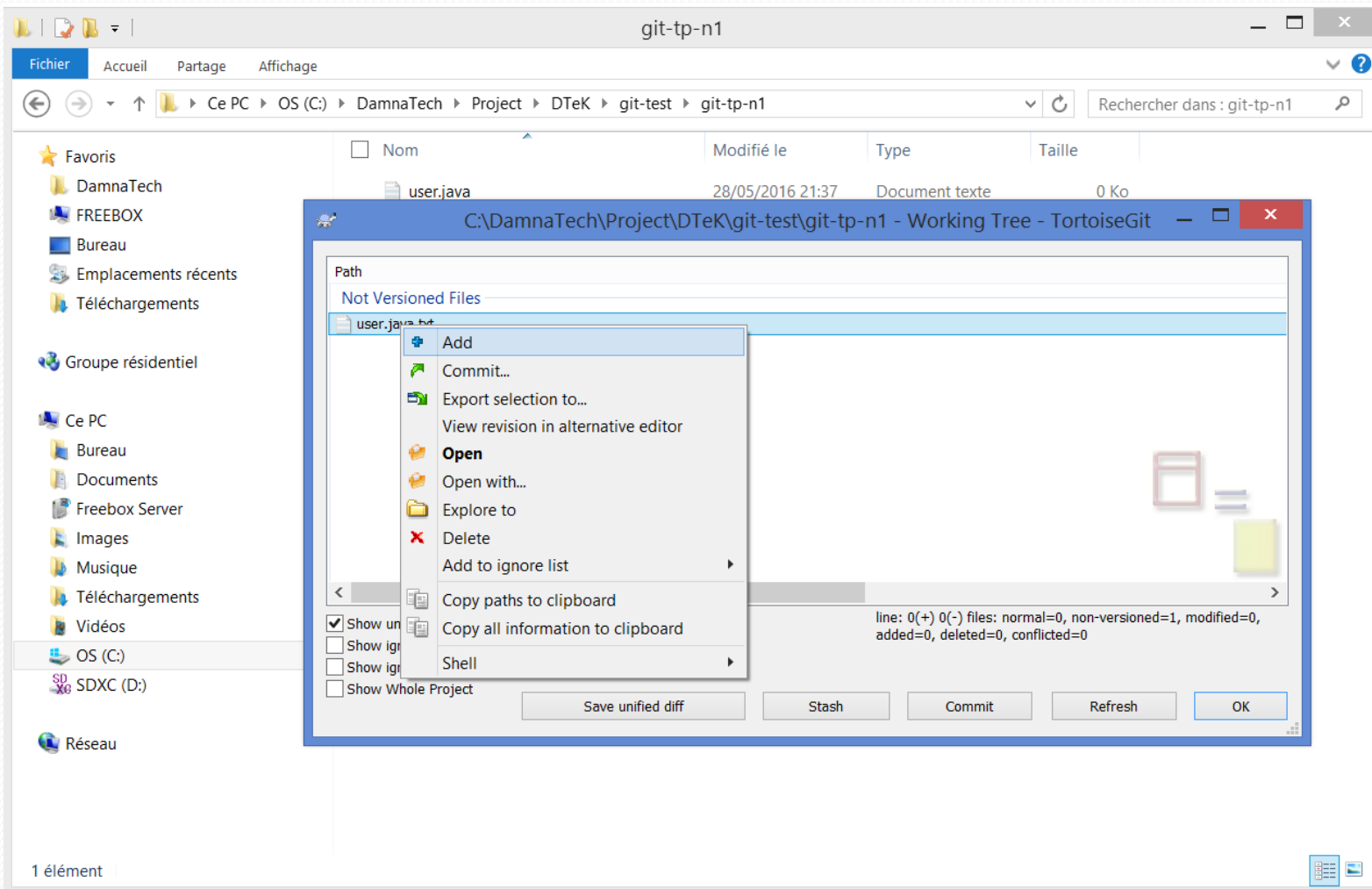
# IV. TortoiseGit

## IV.III. Ajout d'un nouveau fichier (1)



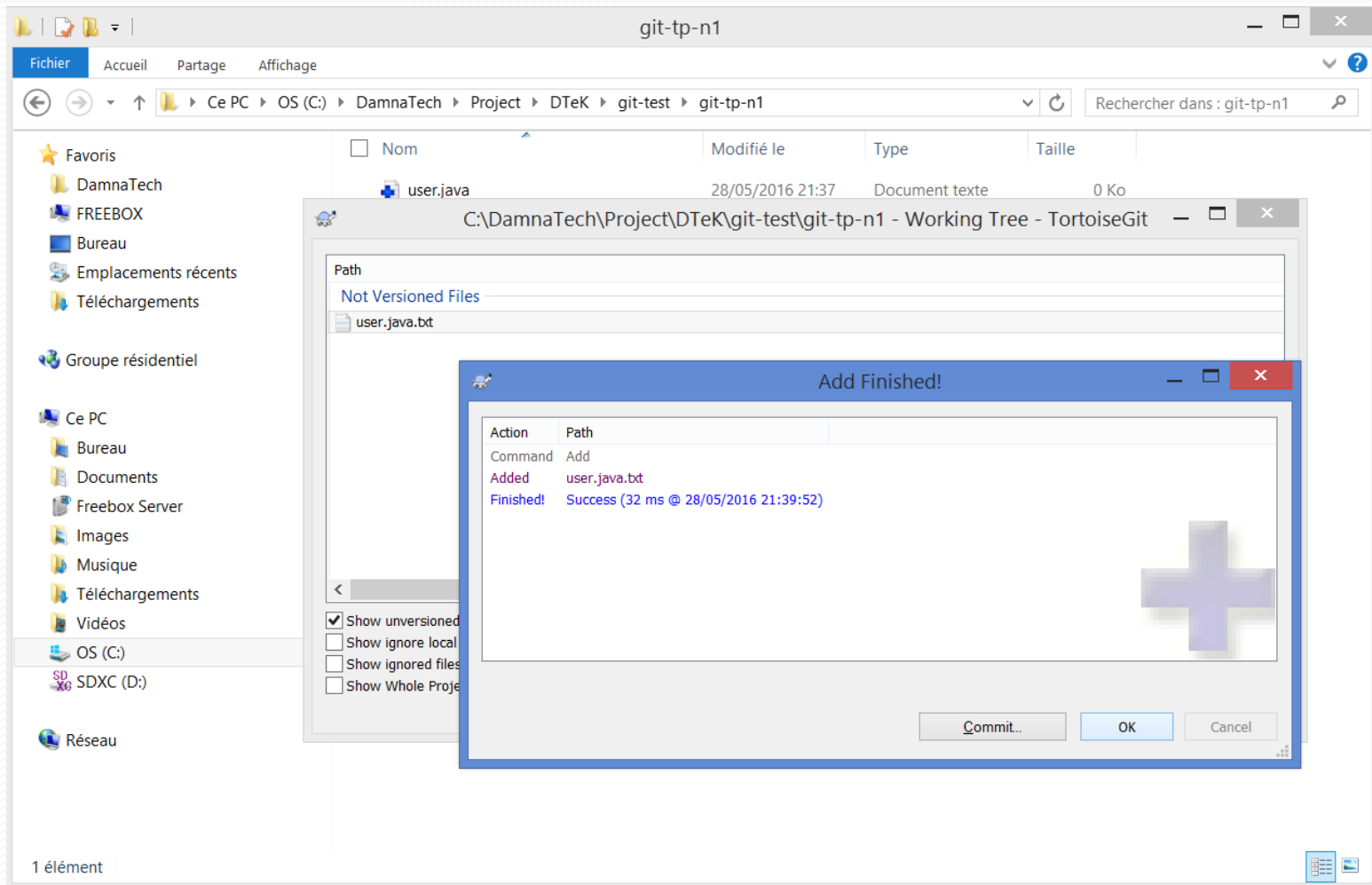
# IV. TortoiseGit

## IV.III. Ajout d'un nouveau fichier (2)



# IV. TortoiseGit

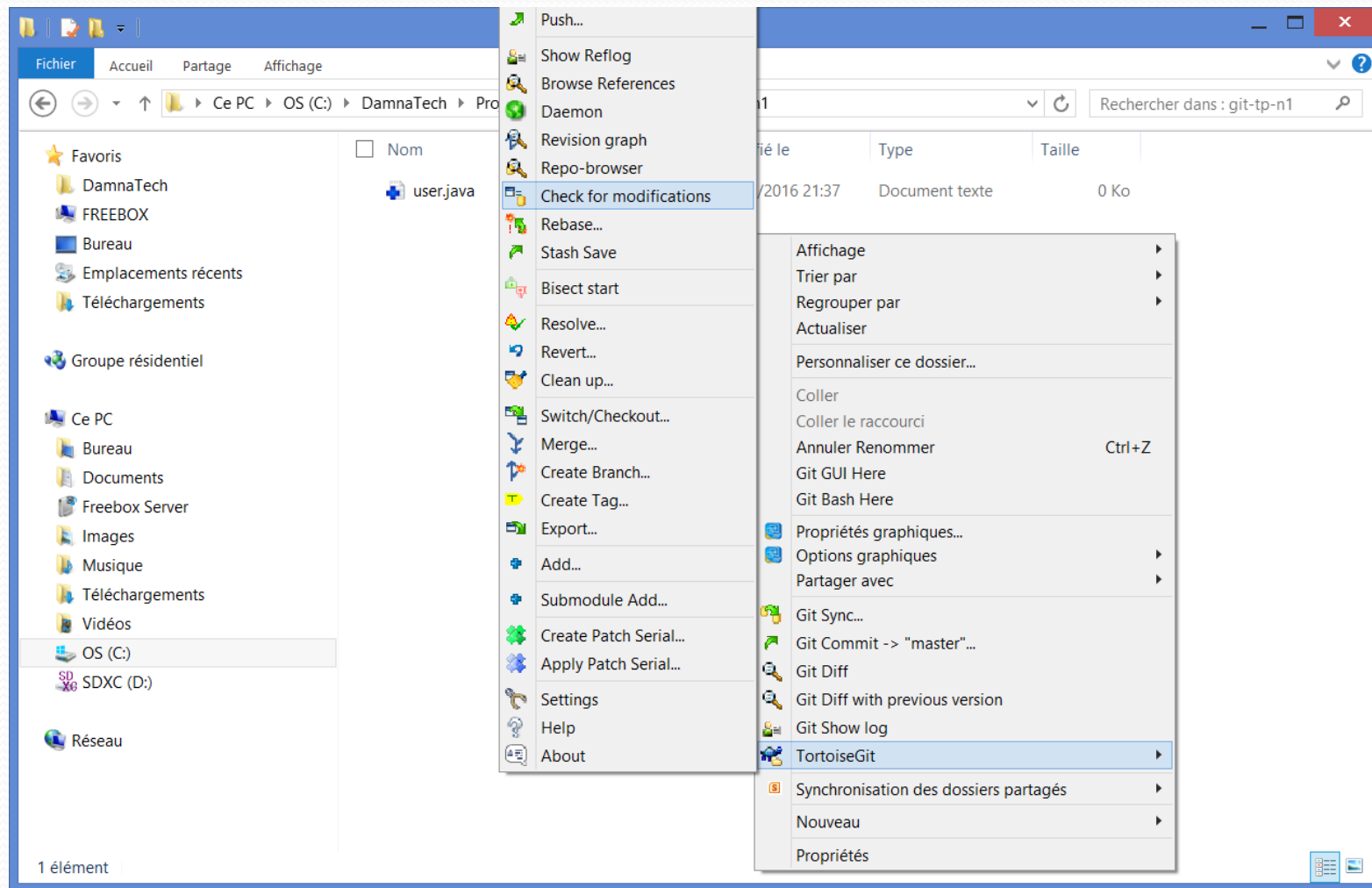
## IV.III. Ajout d'un nouveau fichier (3)





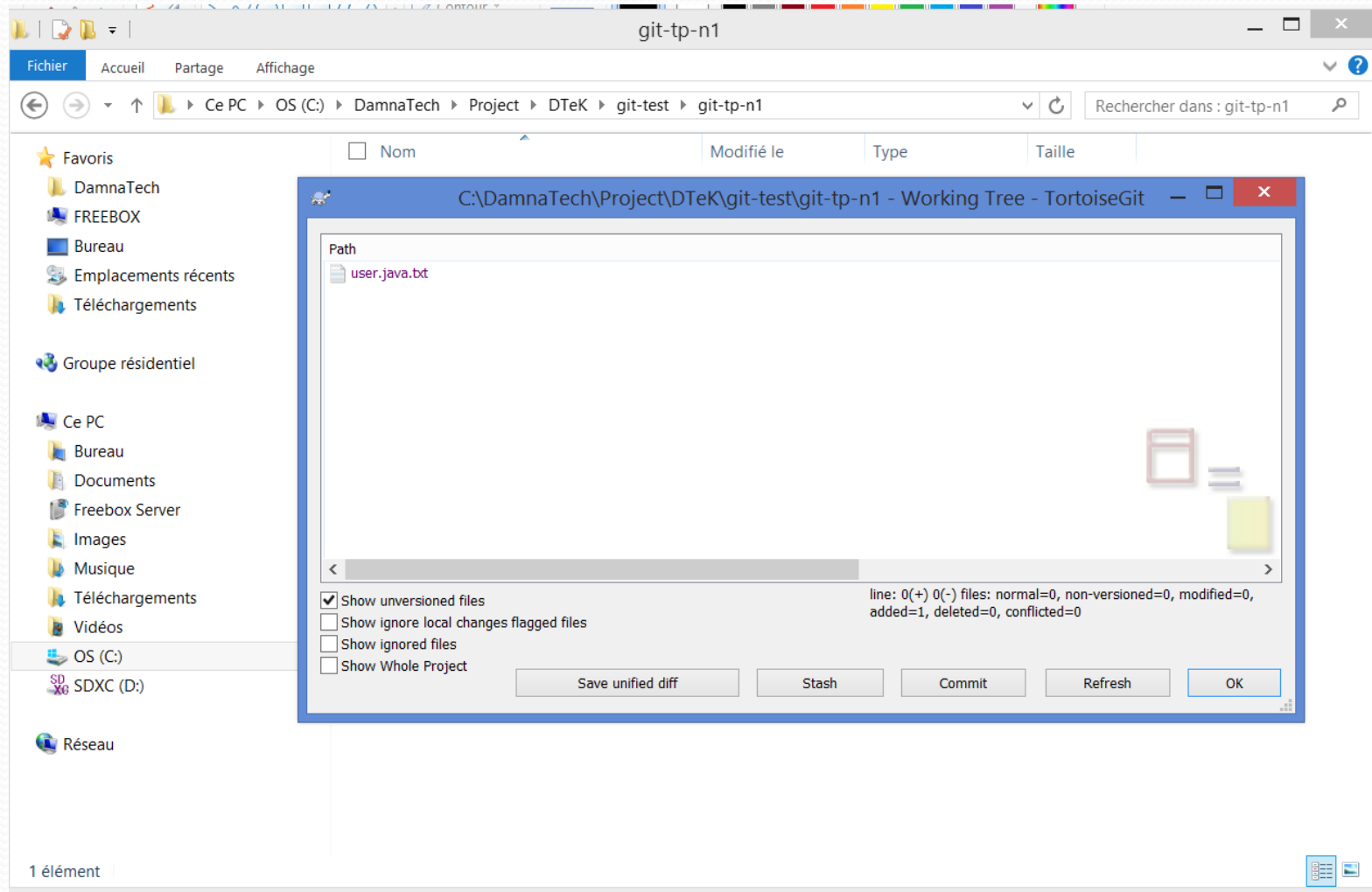
# IV. TortoiseGit

## IV.IV. Publication des modifications (1)



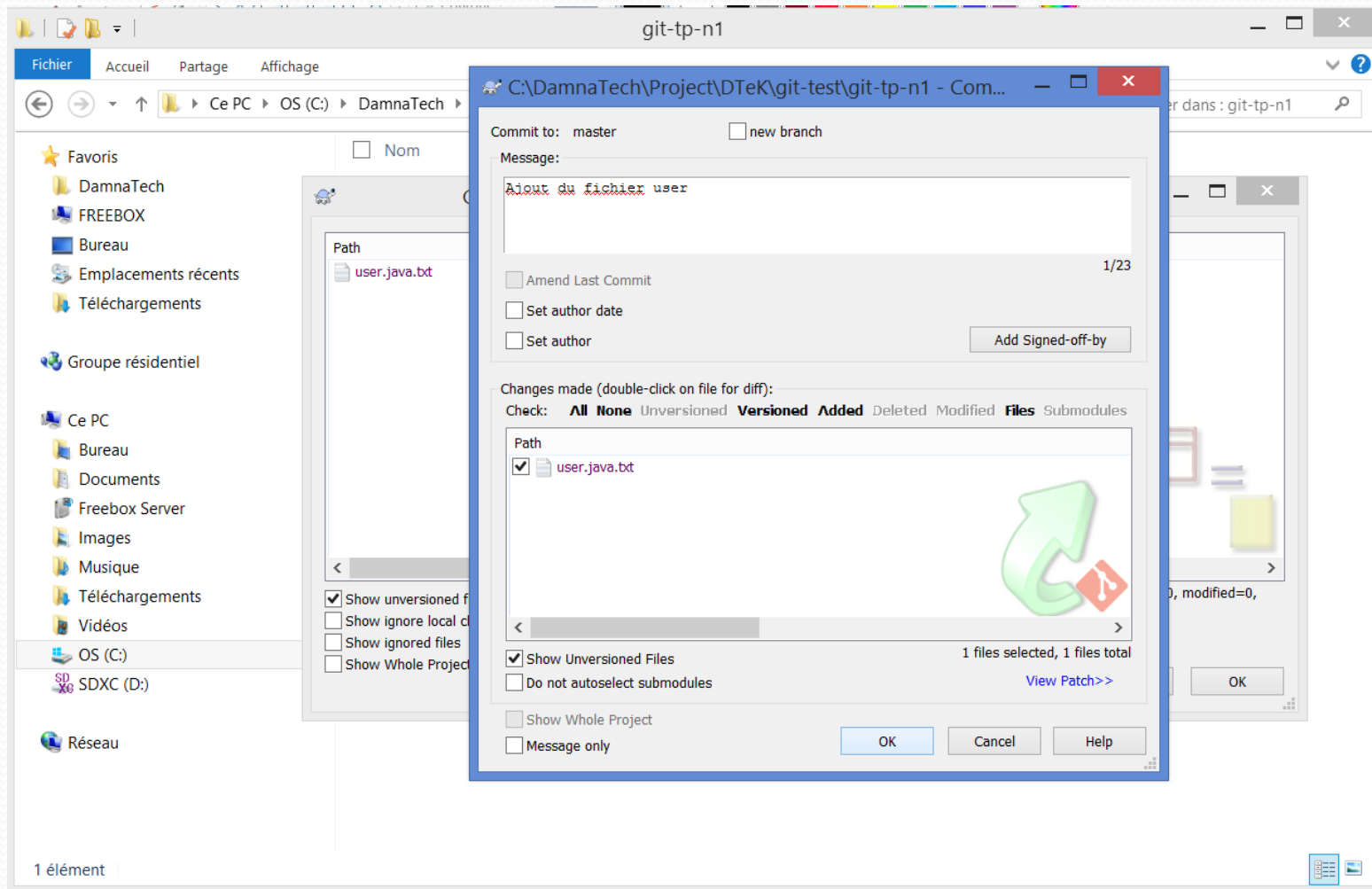
# IV. TortoiseGit

## IV.IV. Publication des modifications (2)



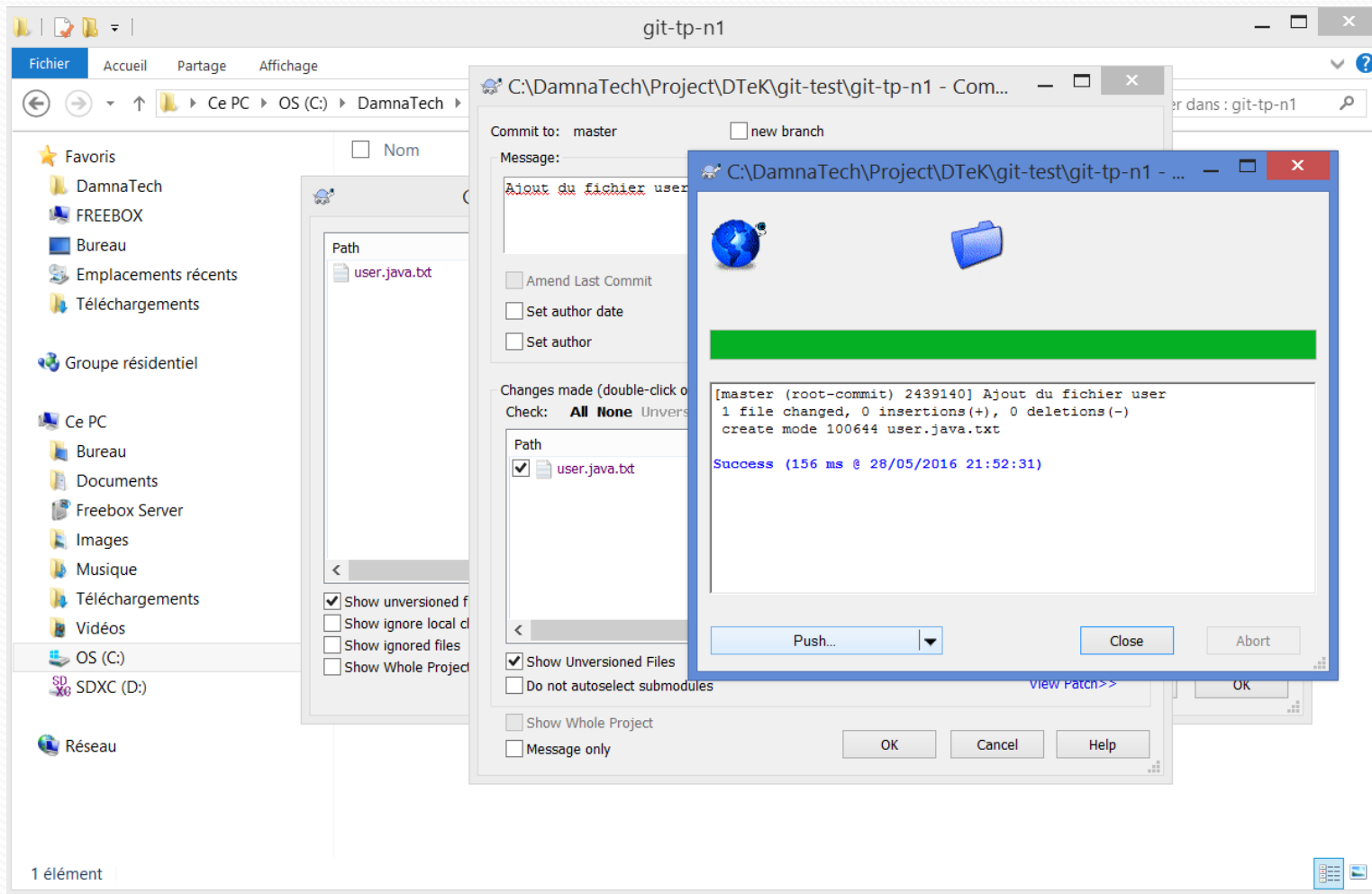
# IV. TortoiseGit

## IV.IV. Publication des modifications (3)



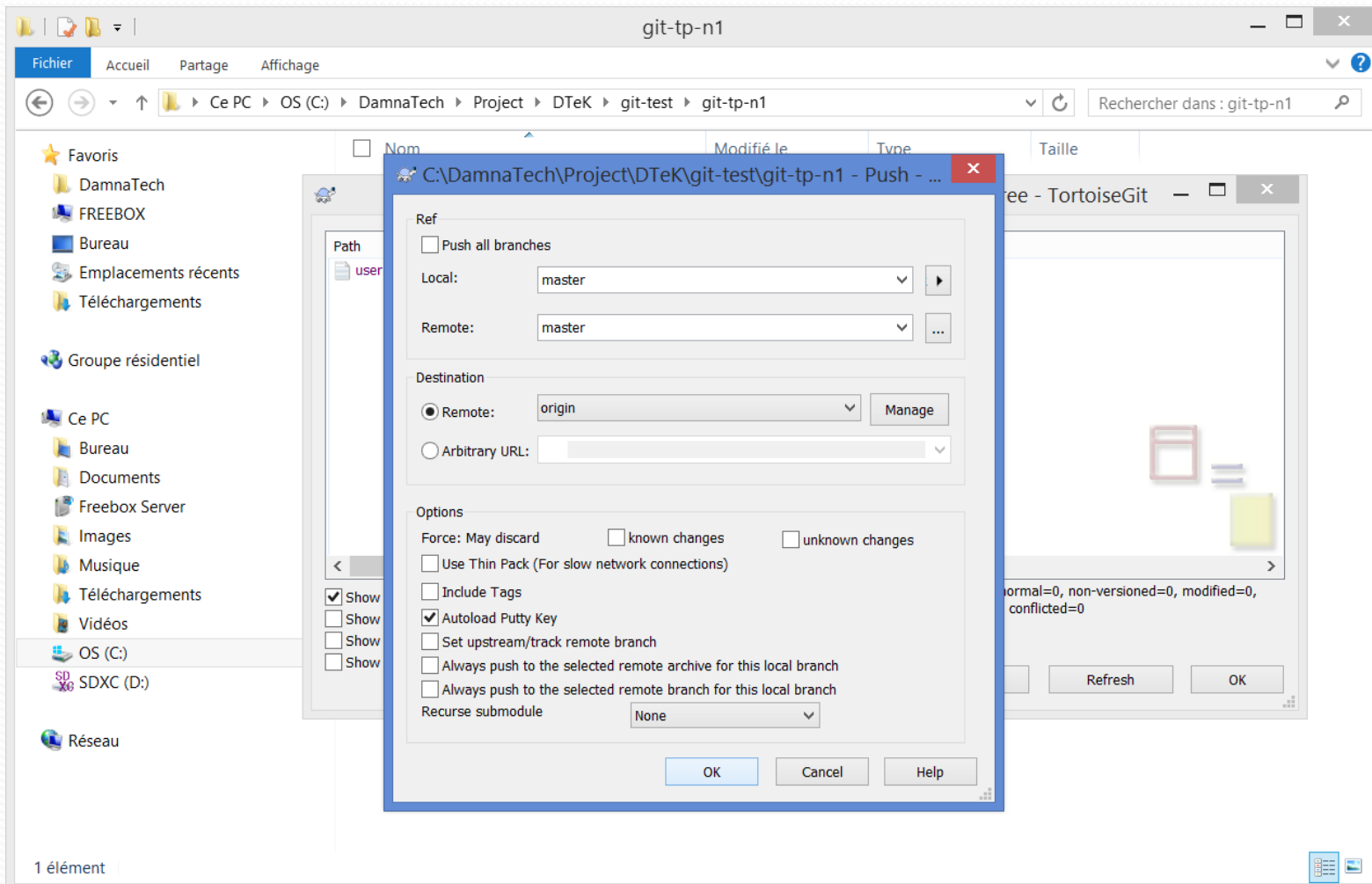
# IV. TortoiseGit

## IV.IV. Publication des modifications (4)



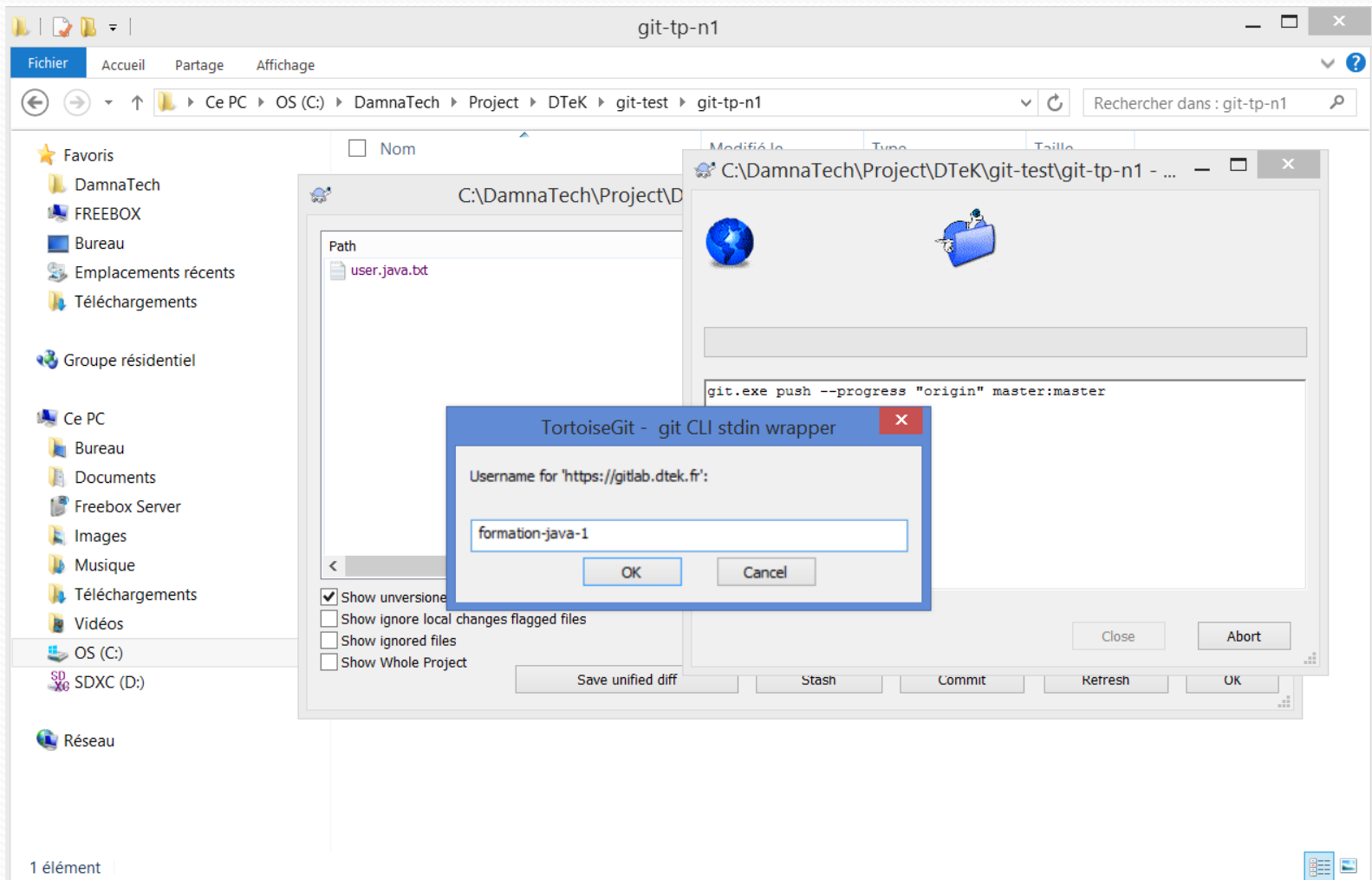
# IV. TortoiseGit

## IV.IV. Publication des modifications (5)



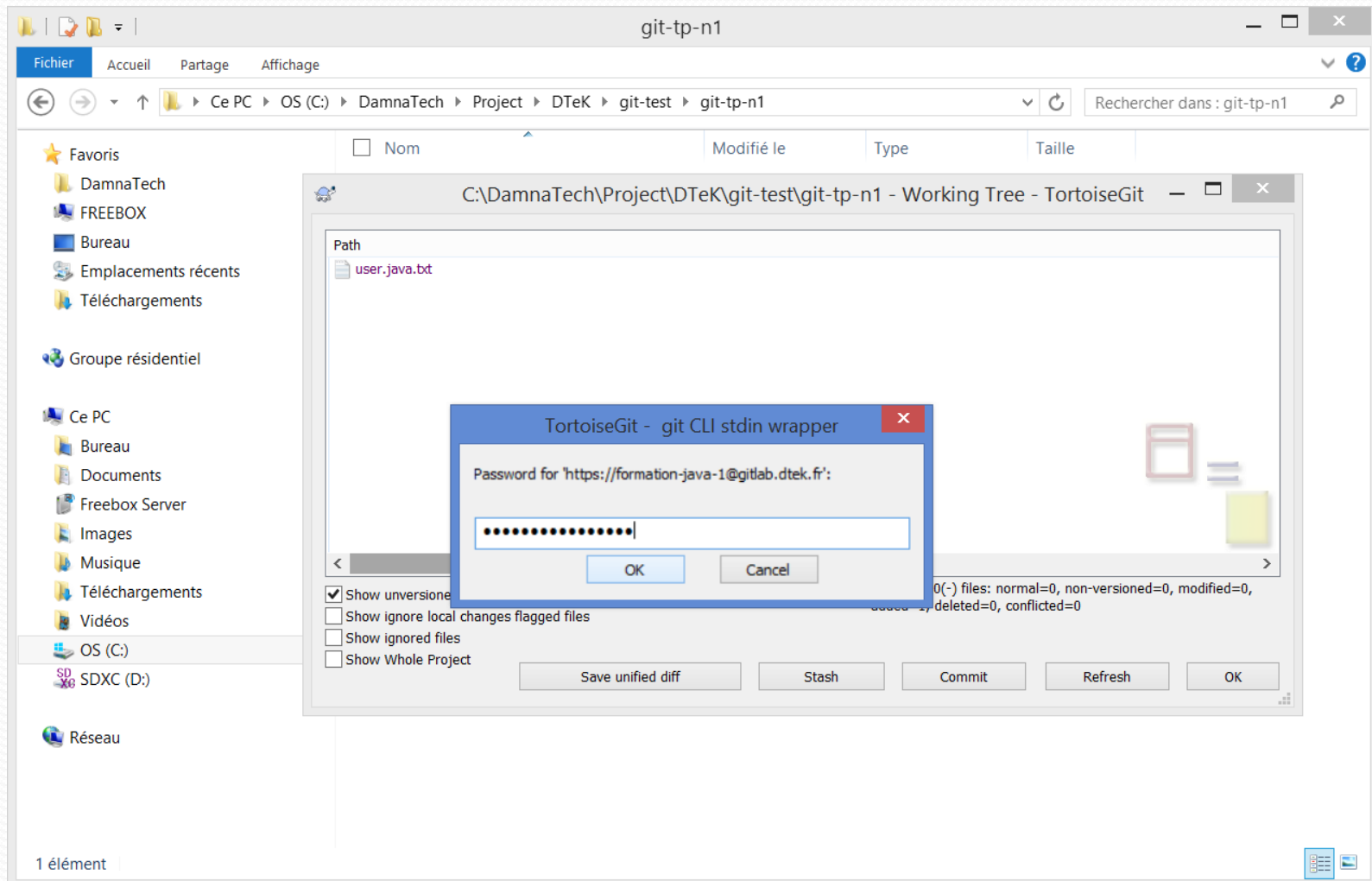
# IV. TortoiseGit

## IV.IV. Publication des modifications (6)



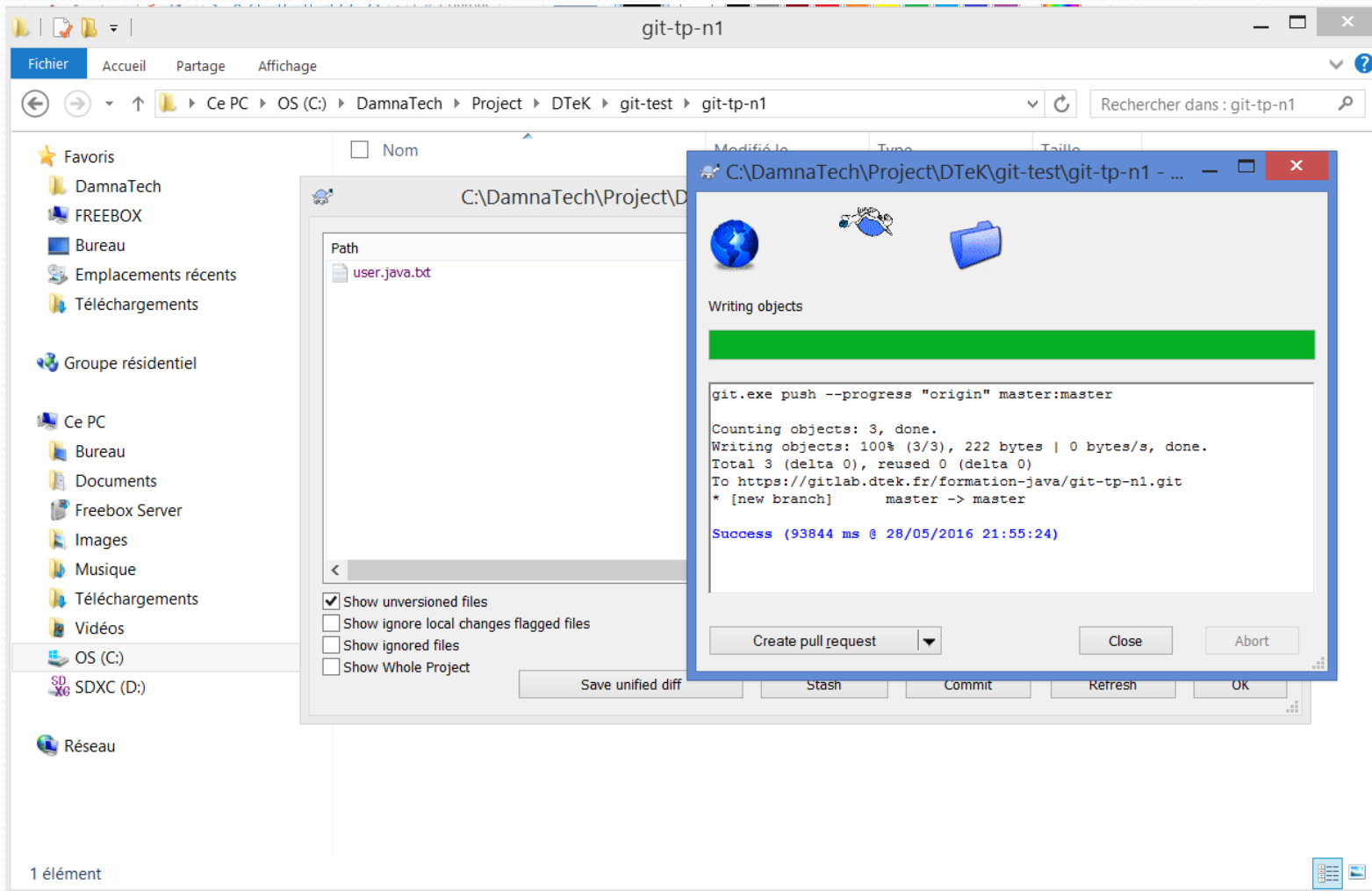
# IV. TortoiseGit

## IV.IV. Publication des modifications (7)



# IV. TortoiseGit

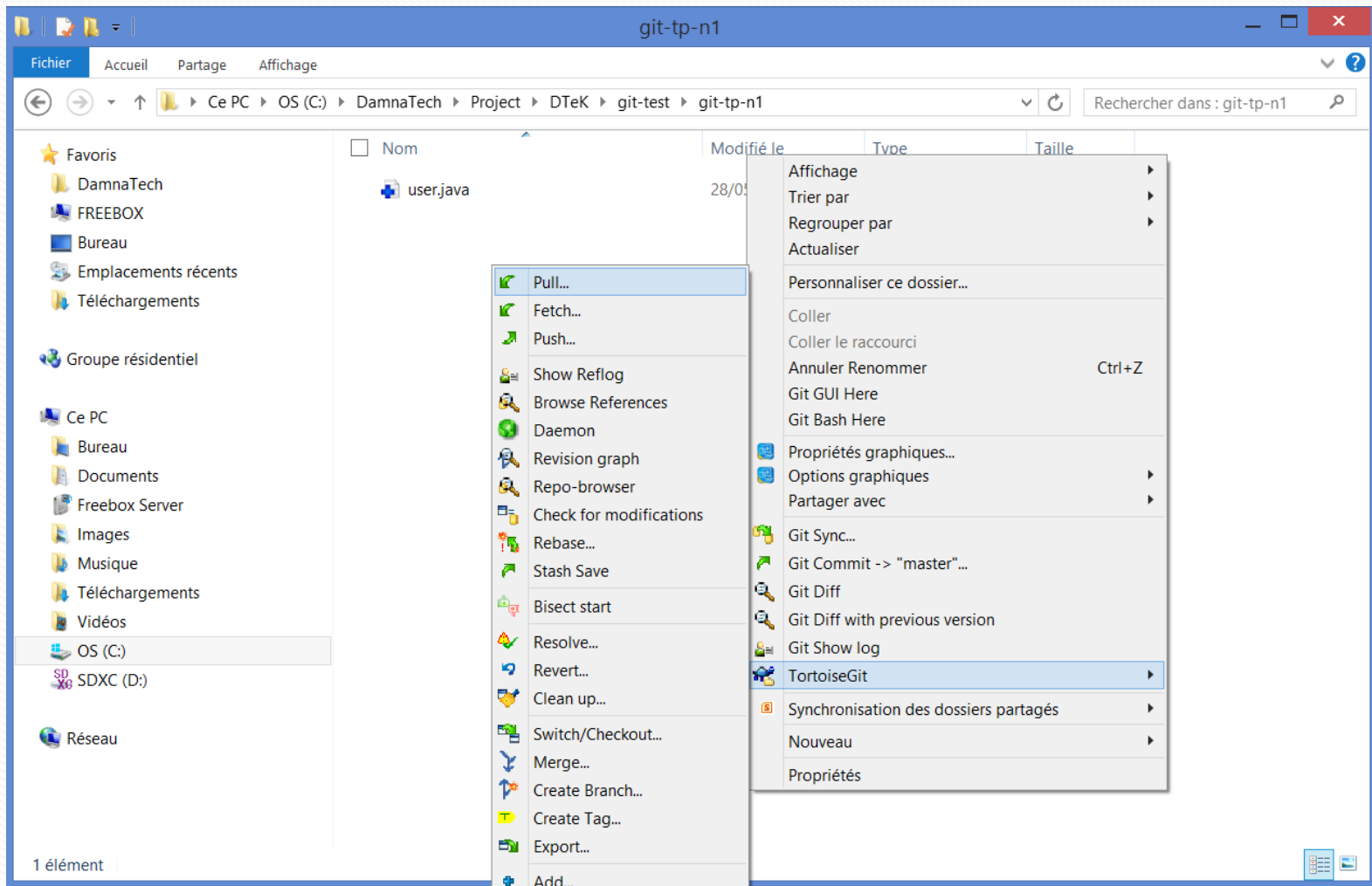
## IV.IV. Publication des modifications (8)





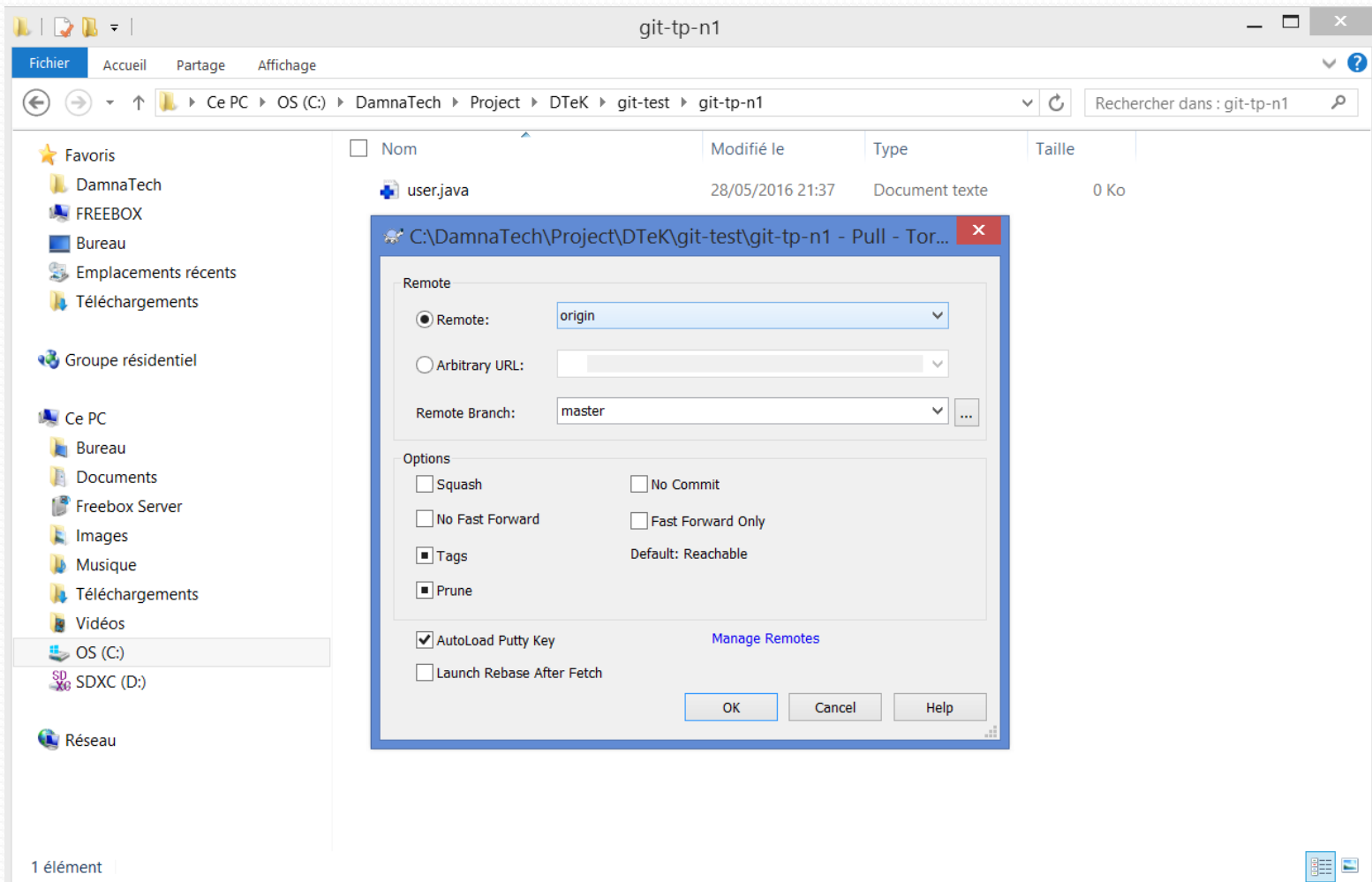
# IV. TortoiseGit

## IV.V. Mise à jour de la version locale (1)



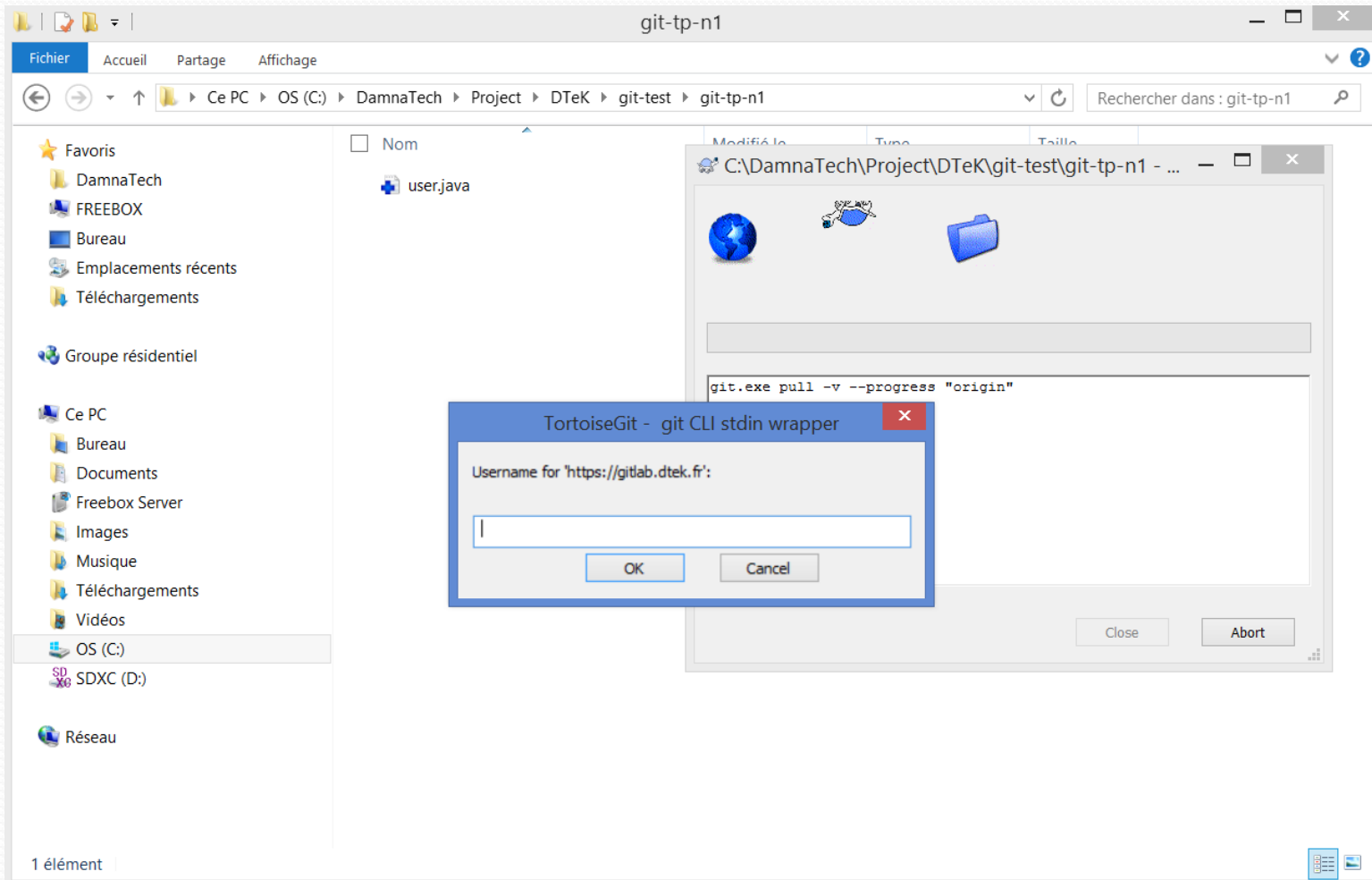
# IV. TortoiseGit

## IV.V. Mise à jour de la version locale (2)



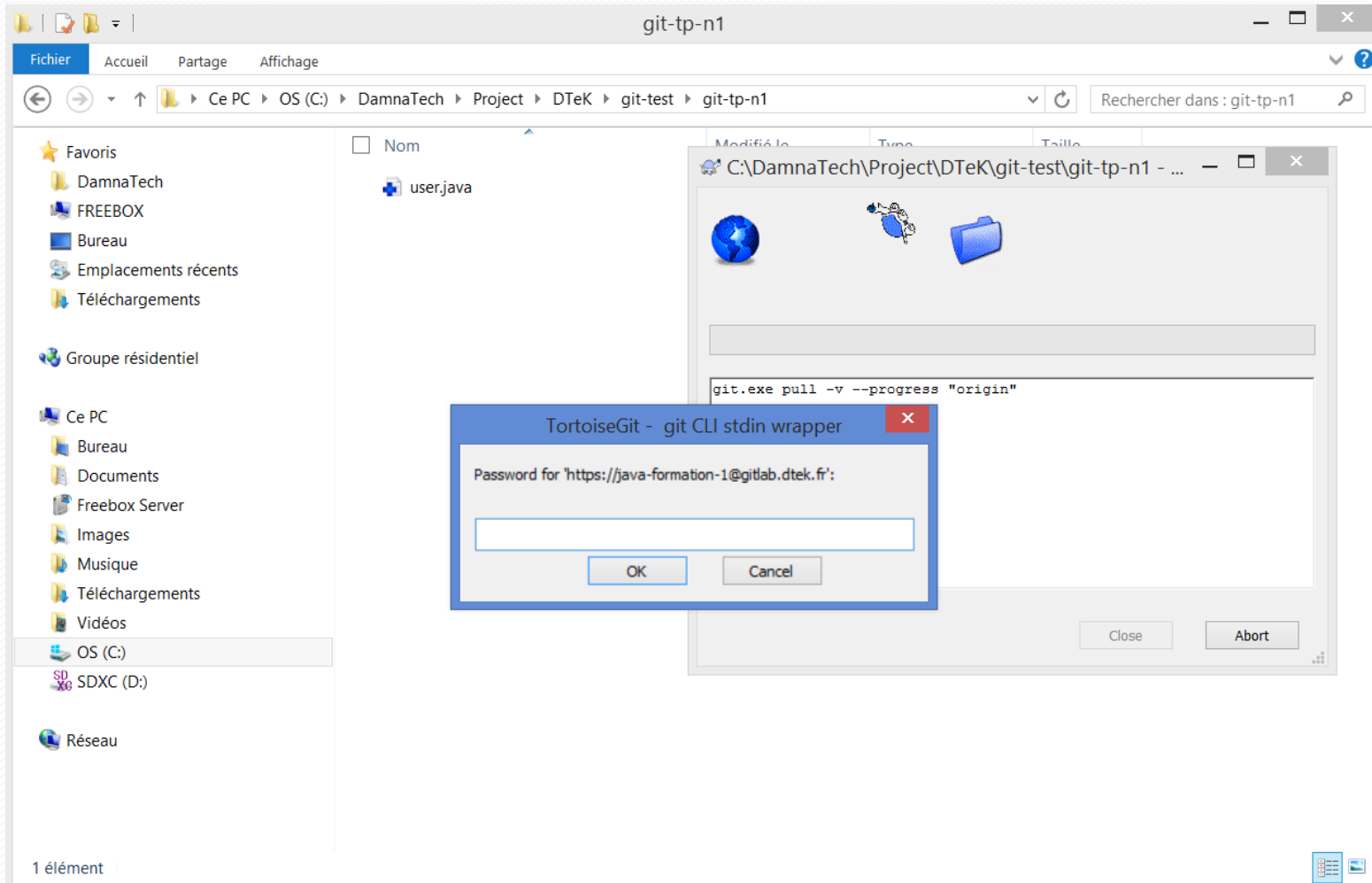
# IV. TortoiseGit

## IV.V. Mise à jour de la version locale (3)



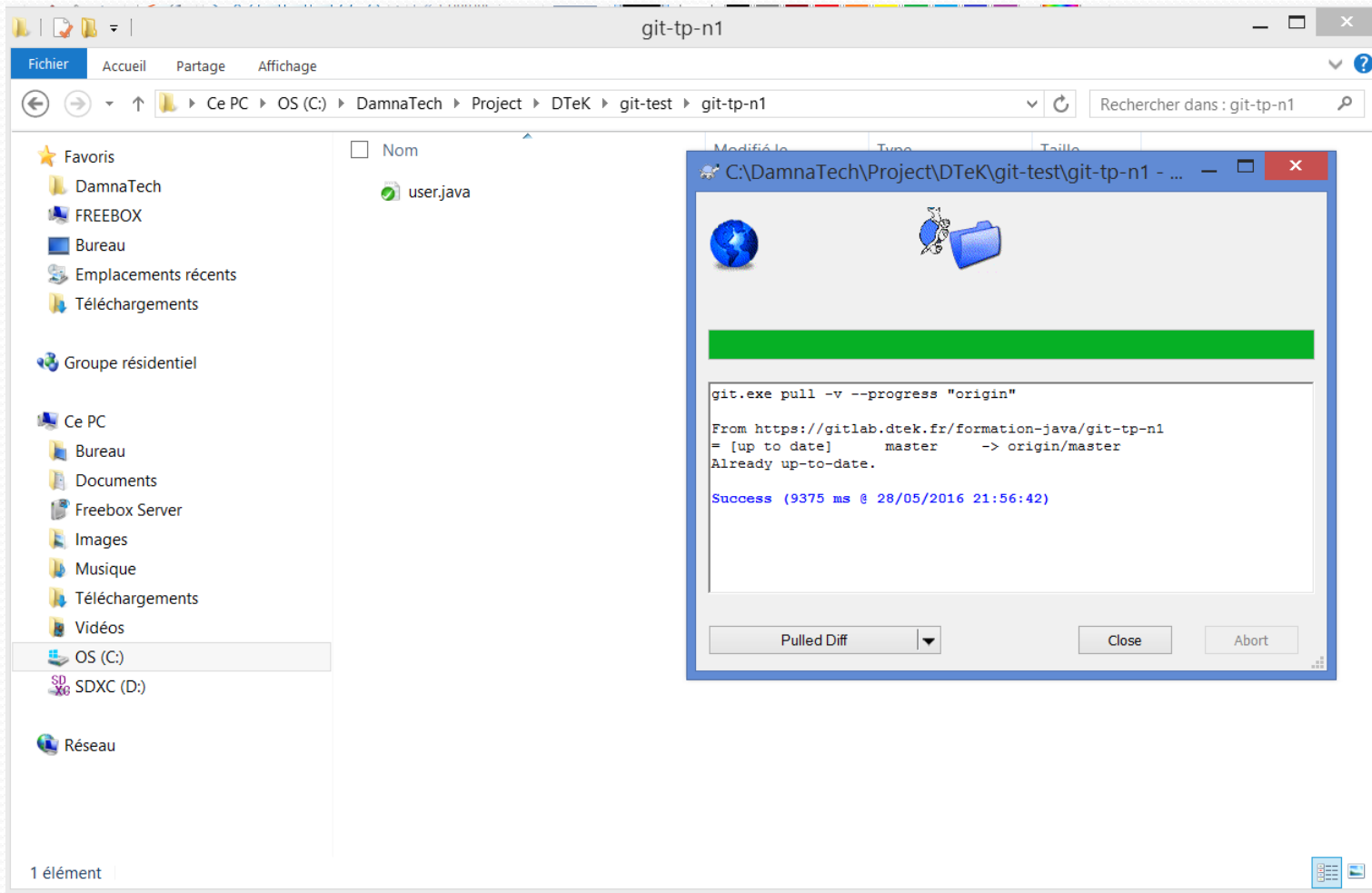
# IV. TortoiseGit

## IV.V. Mise à jour de la version locale (4)



# IV. TortoiseGit

## IV.V. Mise à jour de la version locale (5)



# Merci pour votre attention !!

