Introduction à Git

Mickaël Boiziot

Institut de Recherche en Astrophysique et Planétologie

17 Juin 2016

Introduction - Histoire de Git

- Développé pour le noyeau Linux (150Mo de code source...) par Linus Torvalds
- Créé pour remplacer les autres systèmes de gestion de versions pour Linux
- Première version en avril 2005
- Premier logiciel de gestion de versions par nombre d'utilisateurs (plus de 2 millions)

Introduction

- Git est...
  - gratuit
  - open source
  - très utilisé
  - une technologie pérenne
  - distribué
  - rapide (très très rapide)

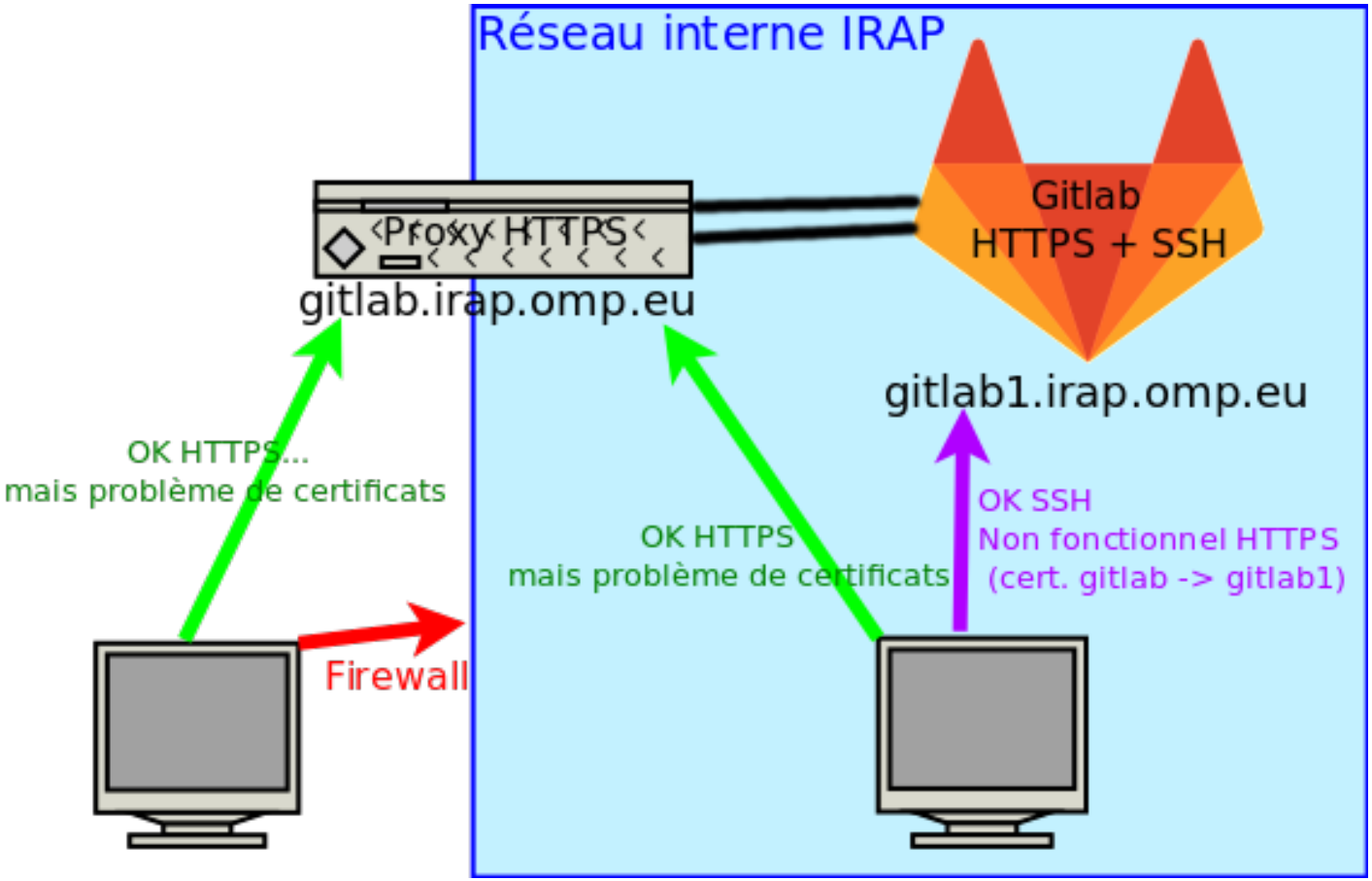
Comparaison avec SVN et CVS

Software	Git	Subversion	CVS
Développement	Actif	Actif	Maintenu
Dépôt	Distribué	Centralisé	Centralisé
Renommage fichier	Oui	Oui	Non
Tag	Oui	Oui	Oui
Branche	Oui	Oui	Oui

Installation

- Linux : utiliser le package manager de votre distribution
- Mac OS X : https ://git-scm.com/downloads
- Windows : https ://git-scm.com/downloads

Git à l'IRAP : Gitlab



Git à l'IRAP : Gitlab

- Pour résumer :
  - A l'extérieur : utiliser gitlab.irap.omp.eu avec HTTPS
  - Au laboratoire : utiliser au choix...
    - gitlab1.irap.omp.eu avec SSH (plus souple)
    - gitlab.irap.omp.eu avec HTTPS

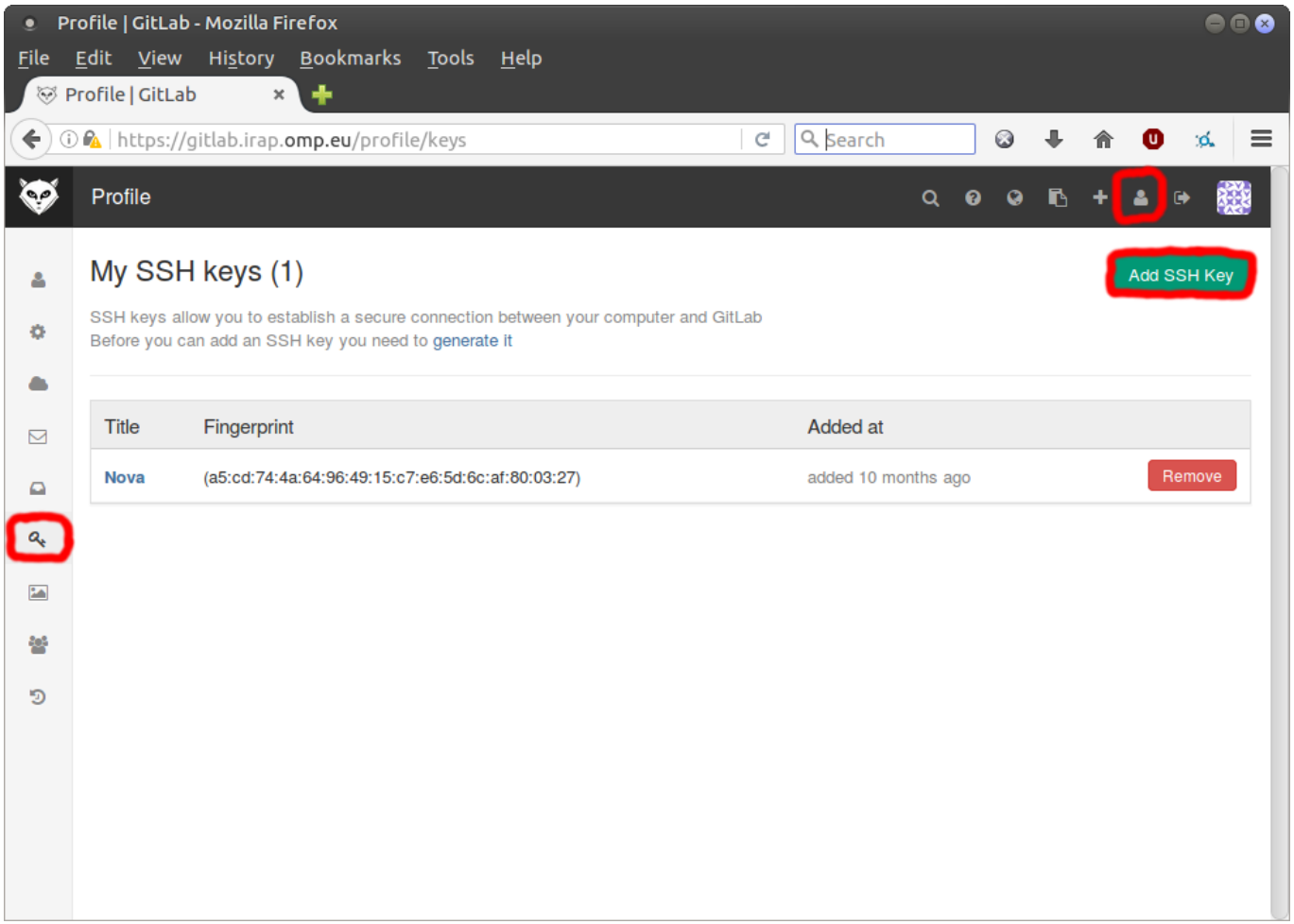
Configuration du client

- Configuration du nom
  git config --global user.name "<Prénom Nom>"
- Configuration de l'email
  git config --global user.email
  "<prenom.nom@irap.omp.eu>"
- Configuration de l'éditeur
  git config --global core.editor <éditeur texte>
- Rendre Git bien plus lisible
  git config --global color.ui auto
- Eviter des commits de merge
  git config --global branch.autosetuprebase always

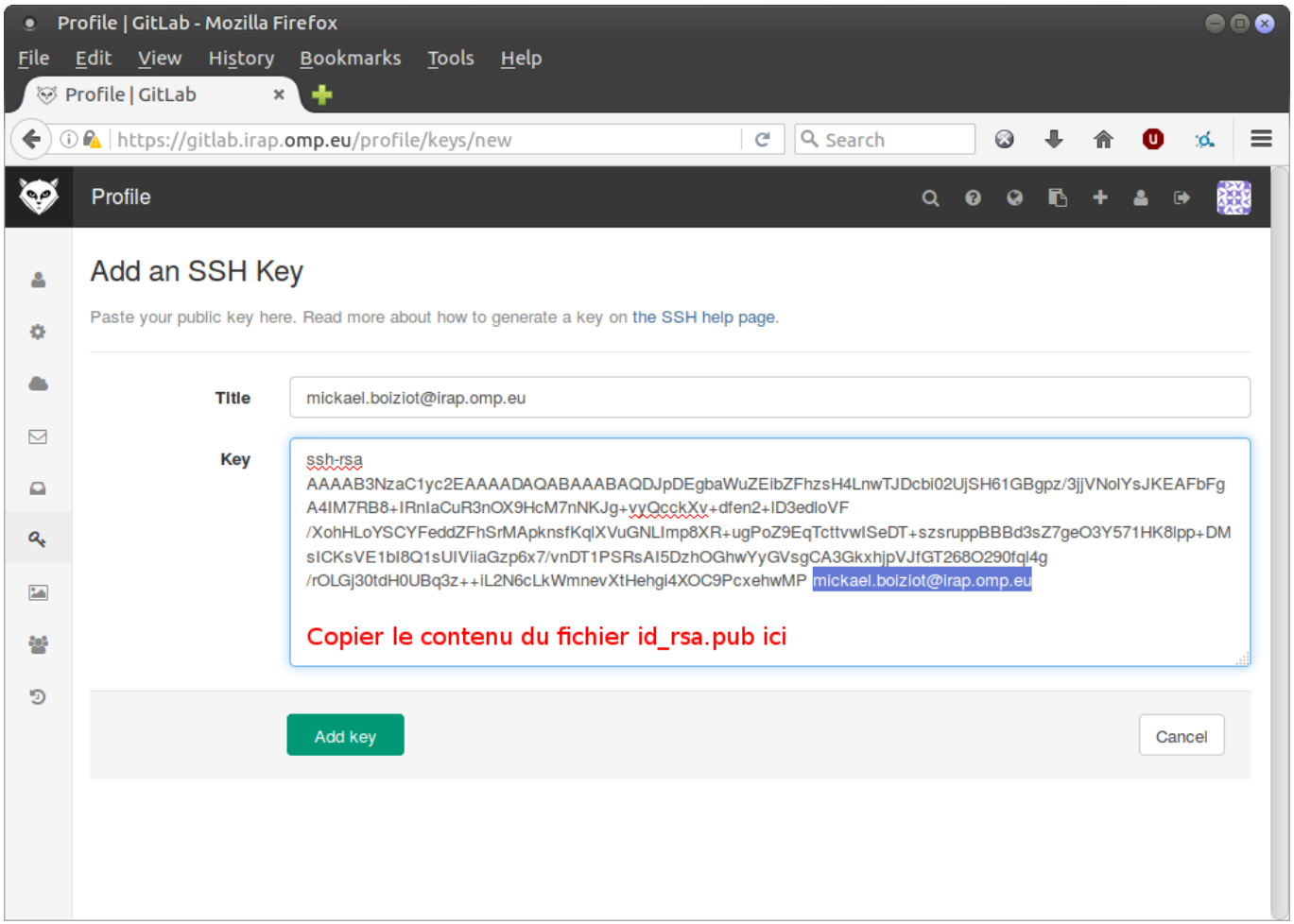
Génération d'une clef SSH

- Linux et Mac OS
  - ssh-keygen -t rsa -C "<IDENTIFIANT>"
  - Clefs créées dans ~/.ssh/
- Windows
  - ssh-keygen.exe -t rsa -C "<IDENTIFIANT>"
  - Clefs créées dans C:\Users\<USER>\.ssh\
- Au moins deux fichiers :
  - Partie privée de la clef : id_rsa
  - Partie public de la clef : id_rsa.pub

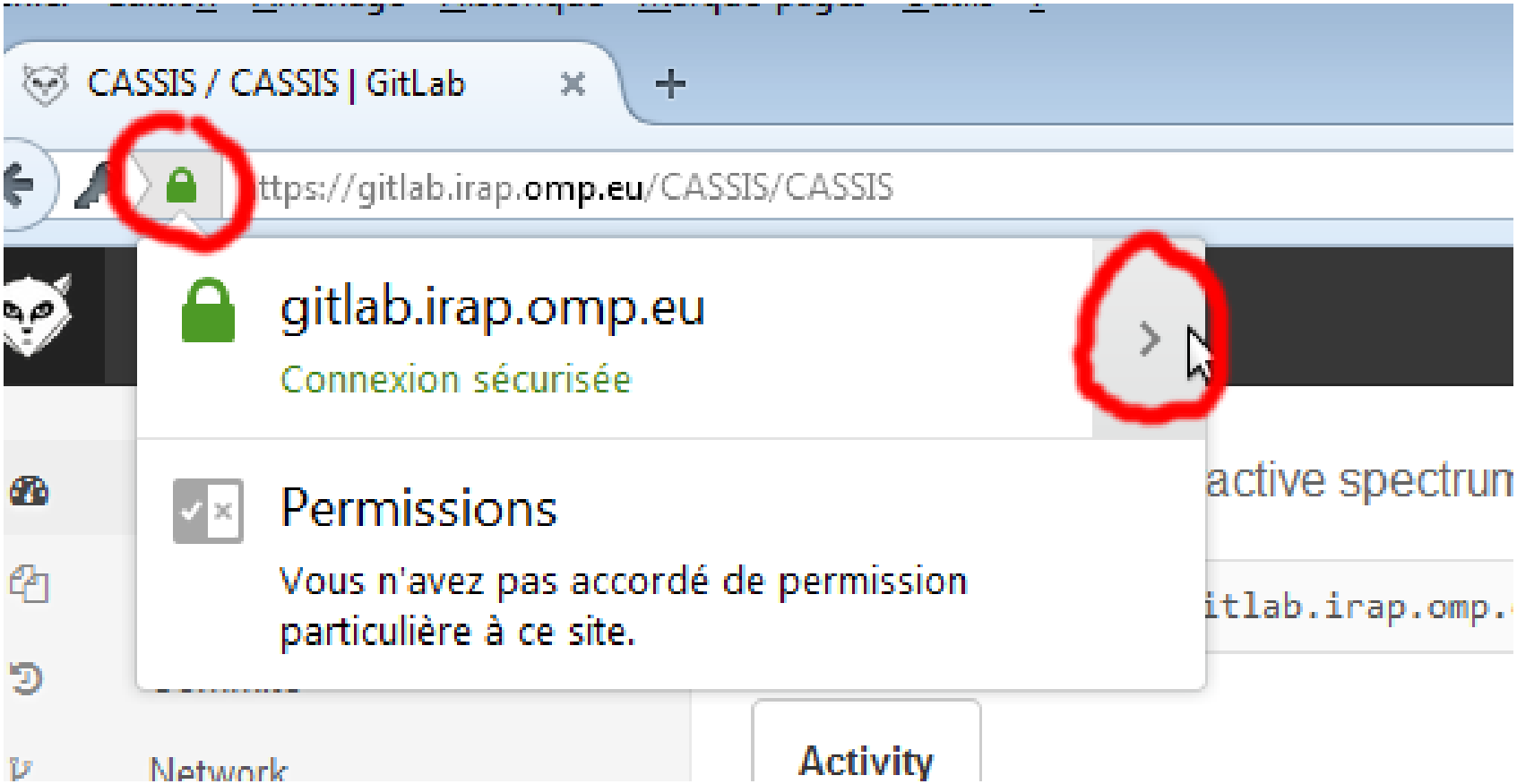
Installation dans Gitlab - 1



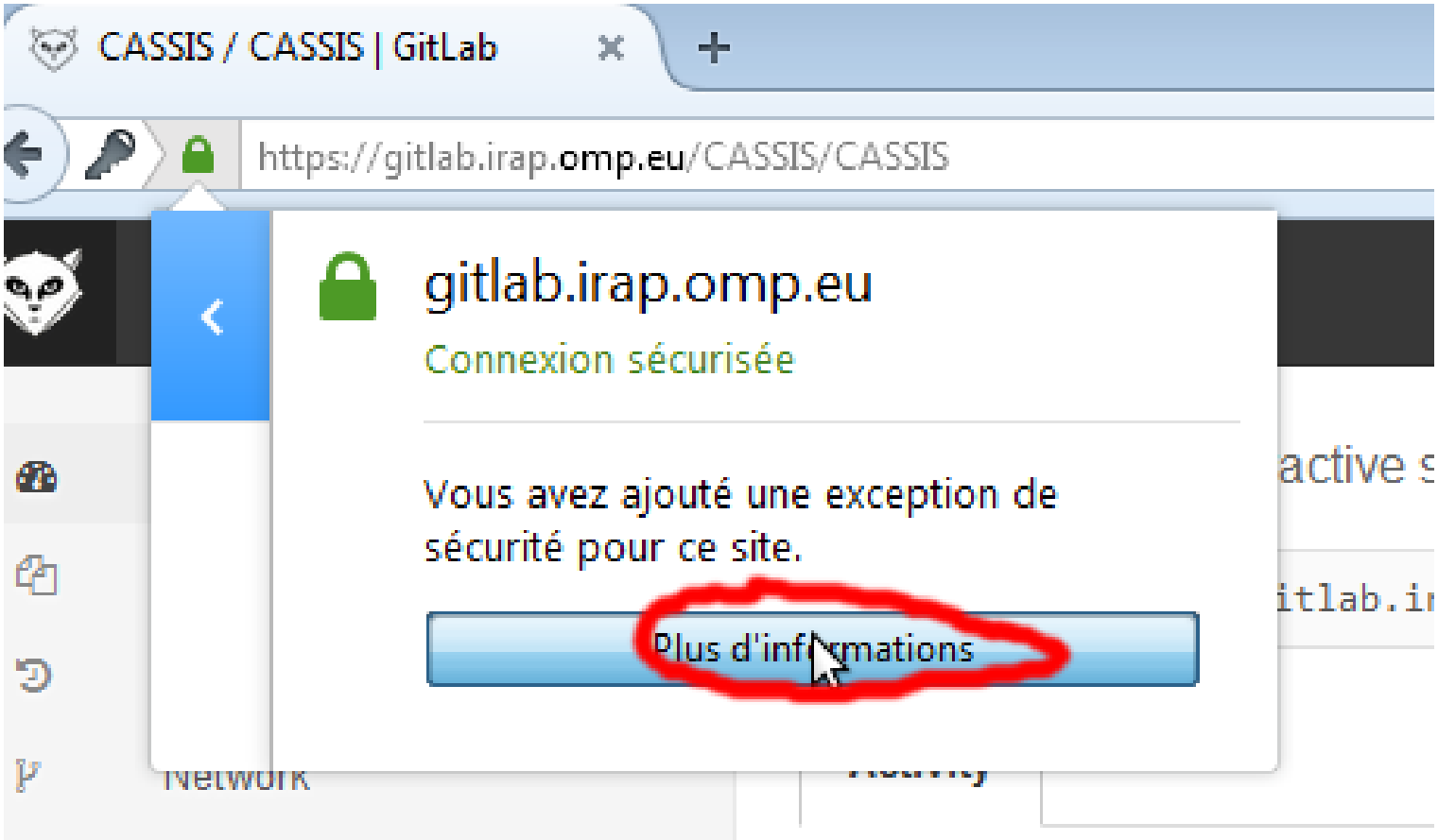
Installation dans Gitlab - 2



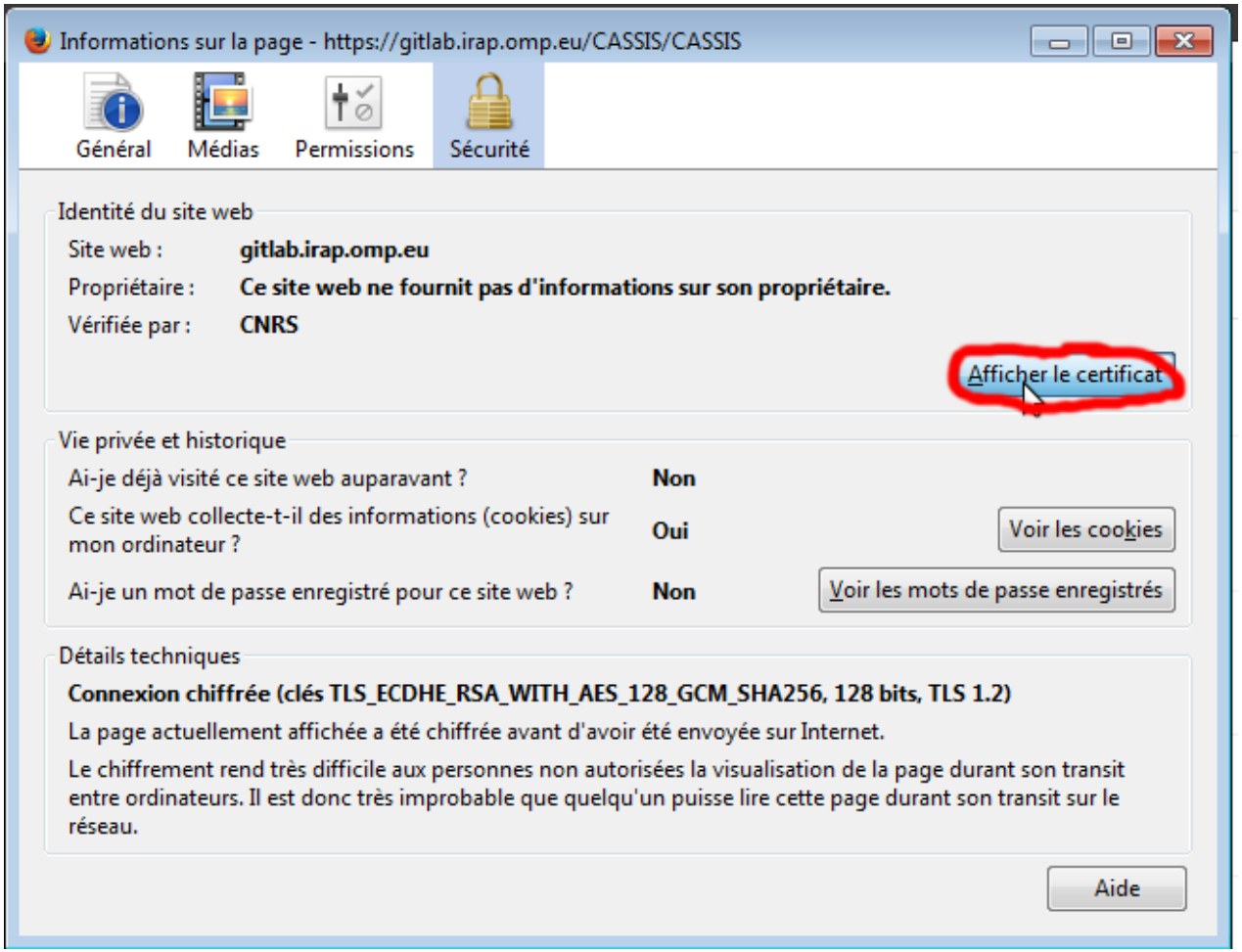
Récupération Certificat - 1



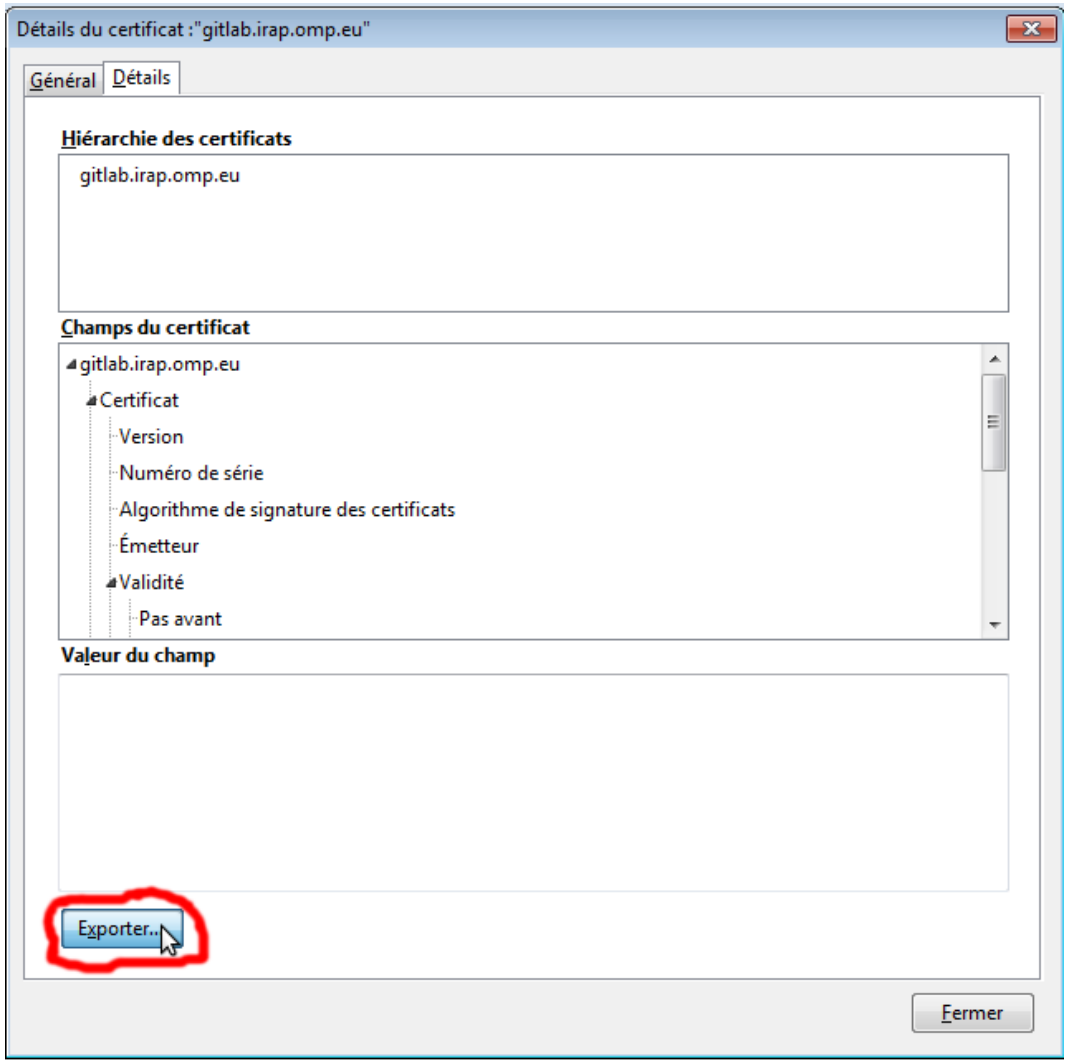
Récupération Certificat - 2



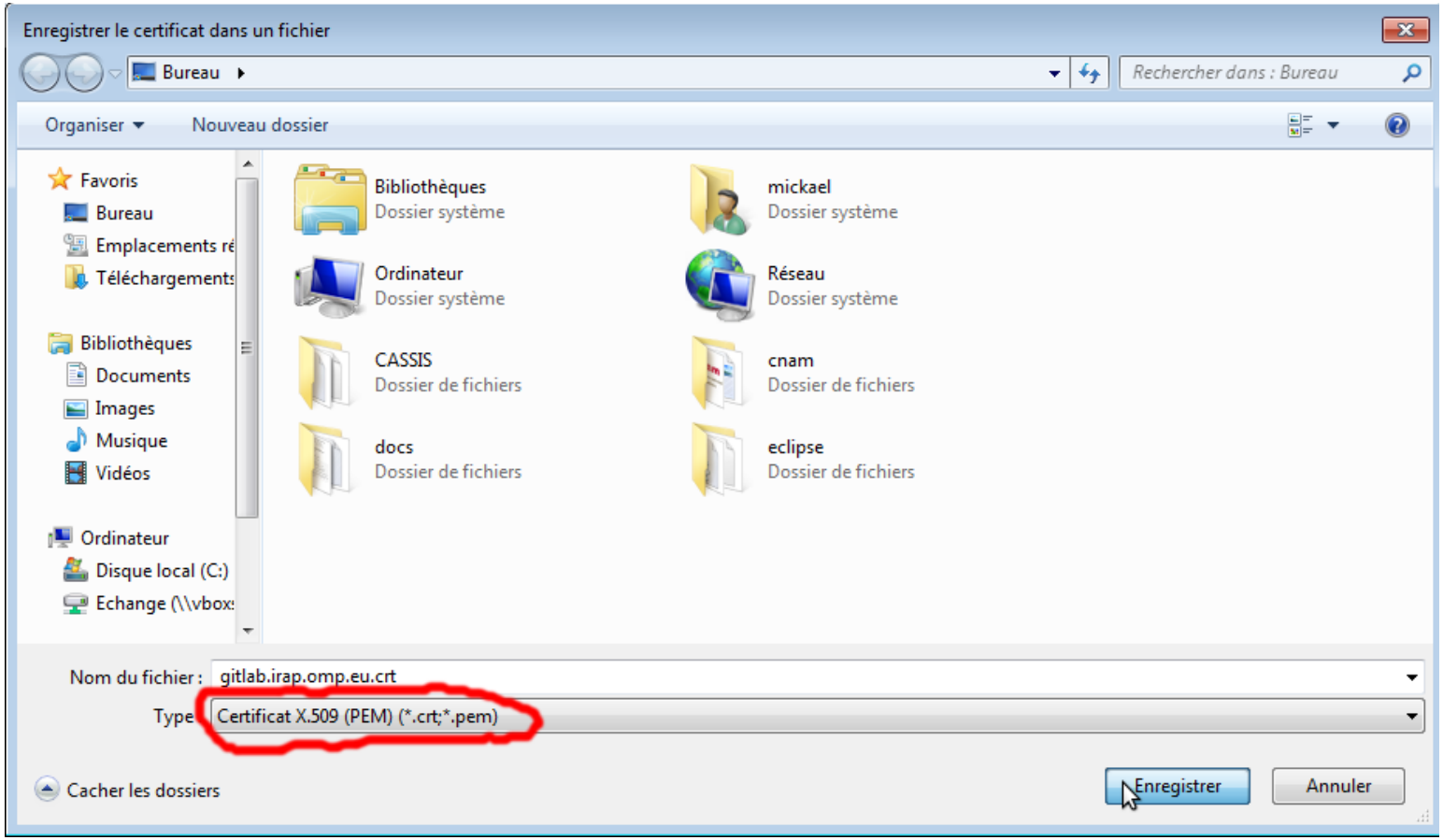
Récupération Certificat - 3



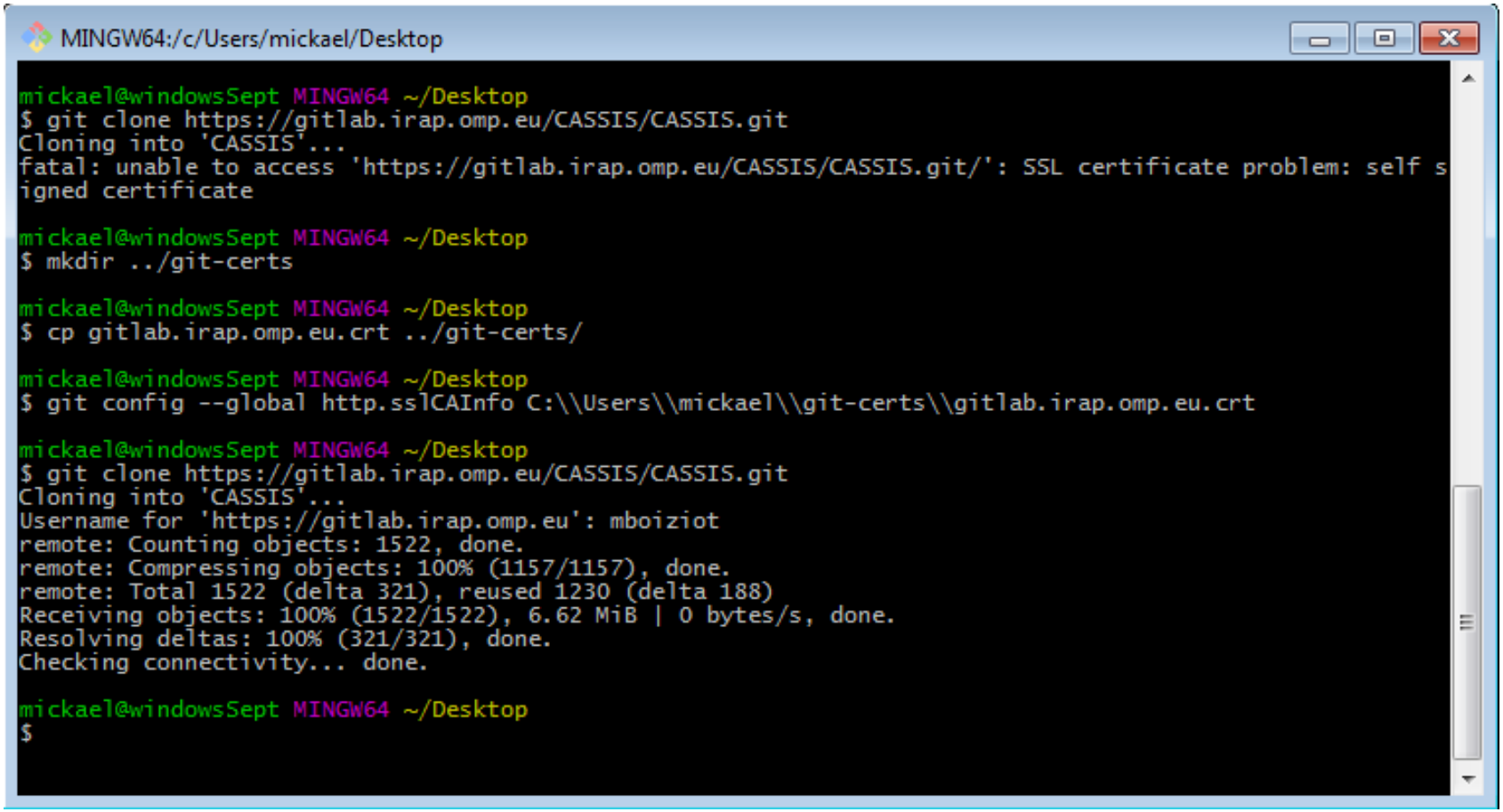
Récupération Certificat - 4



Récupération Certificats - 5



Installation sous windows - 1



Installation sous windows - 2

```
mkdir ../git-certs
cp gitlab.irap.omp.eu.crt ../git-certs/
git config --global http.sslCAInfo C:\\Users\\mickael\\
    git-certs\\gitlab.irap.omp.eu.crt
```

Installation sous Mac OS

- Cliquer sur le fichier gitlab.irap.omp.eu.crt
- Choisir de l'installer au niveau système

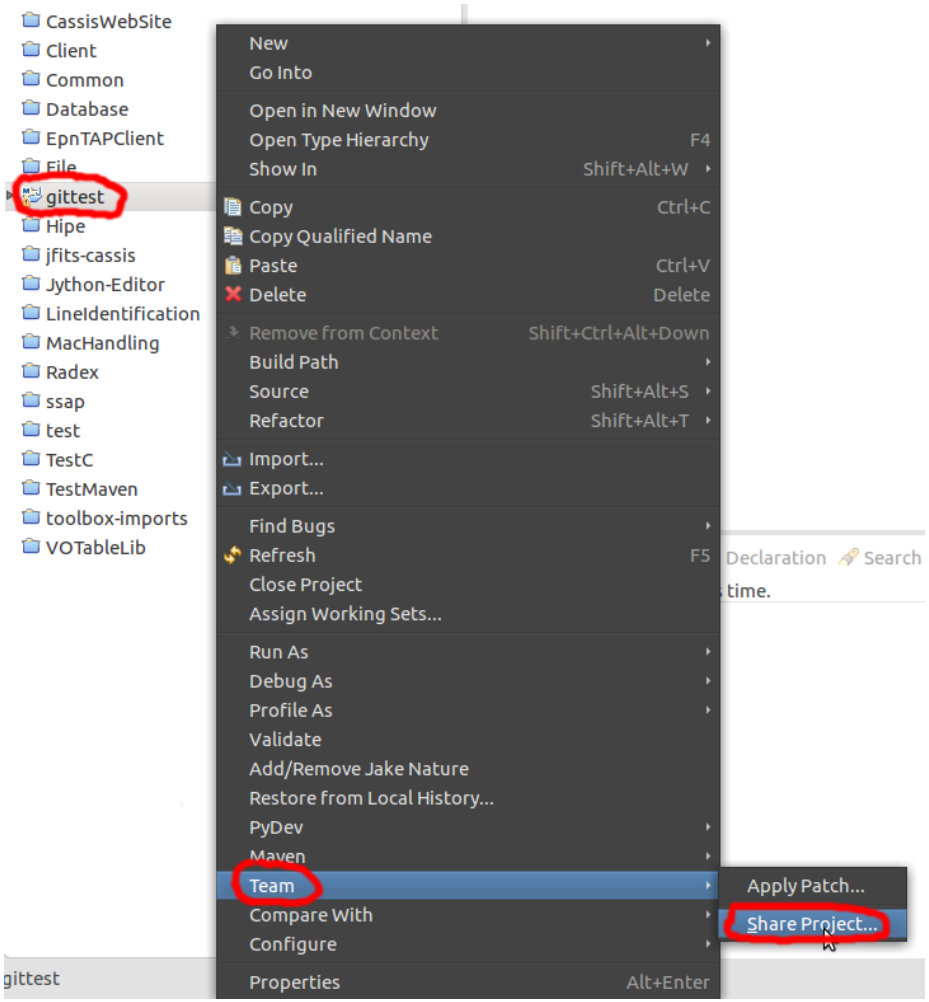
Installation sous Linux (Ubuntu)

- Copier le certificat dans /usr/local/share/ca-certificates/ (il doit absolument avoir l'extension .crt)
- Lancer la commande sudo update-ca-certificates

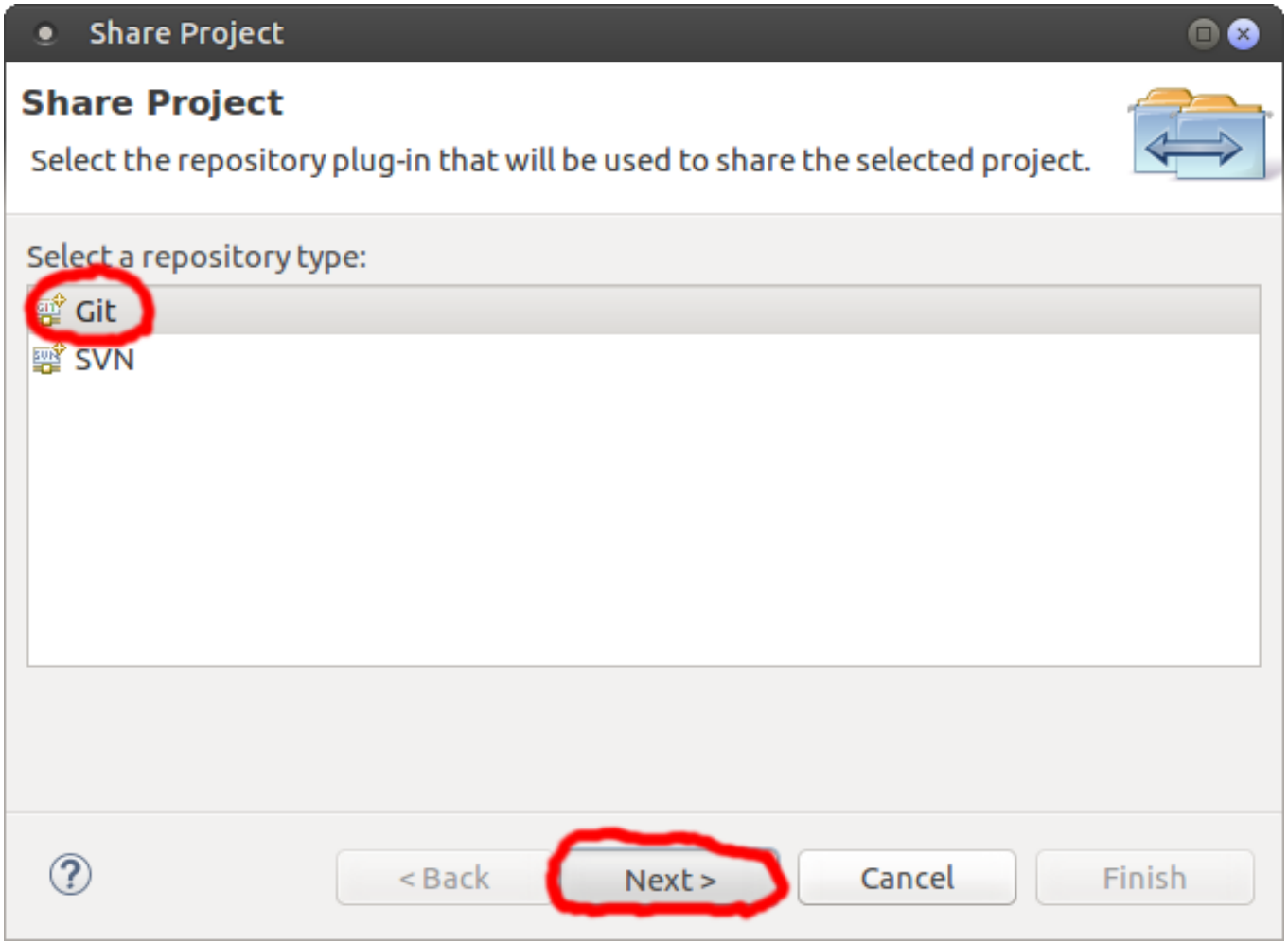
Création d'un dépôt Git

- Se placer dans le dossier du projet
- git init
- Le dépôt Git local est créé : dossier .git

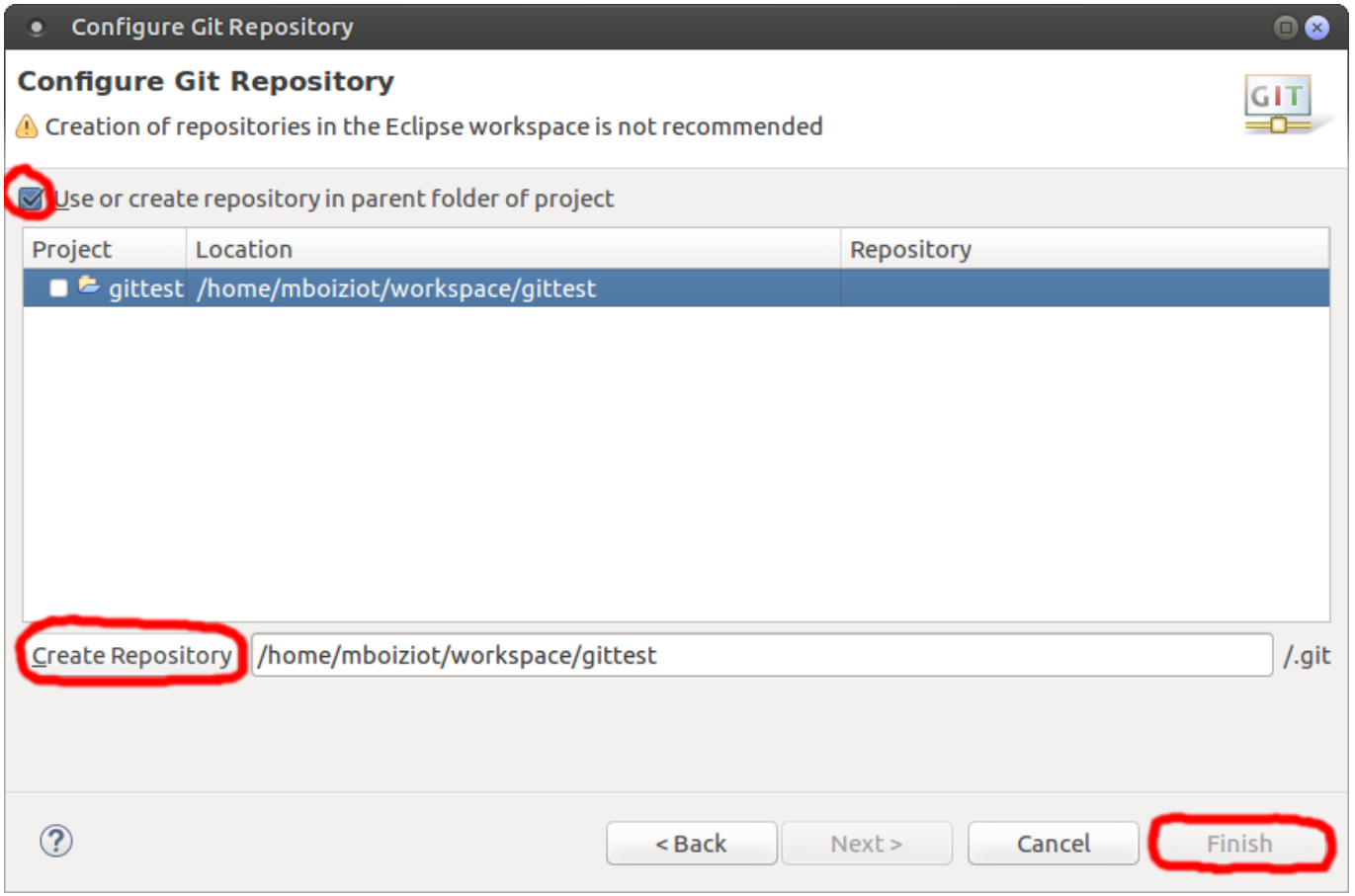
Création d'un dépôt Git avec Eclipse - 1



Création d'un dépôt Git avec Eclipse - 2



Création d'un dépot Git avec Eclipse - 3



Connaître l'état

- git status
  - Informe de la branche courante
  - Informe de l'état par rapport au dépôt distant principal
  - Informe de l'état des fichiers modifiés / nouvellement créés

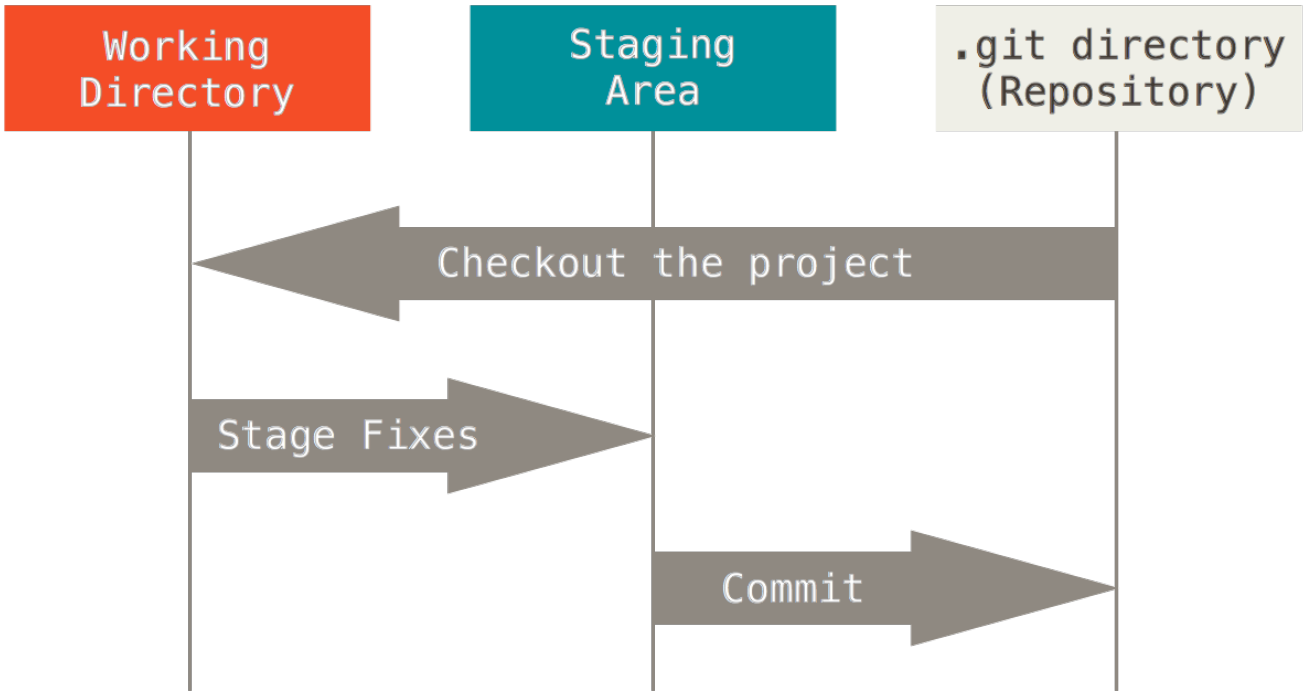
Connaître l'état - exemple

Terminal

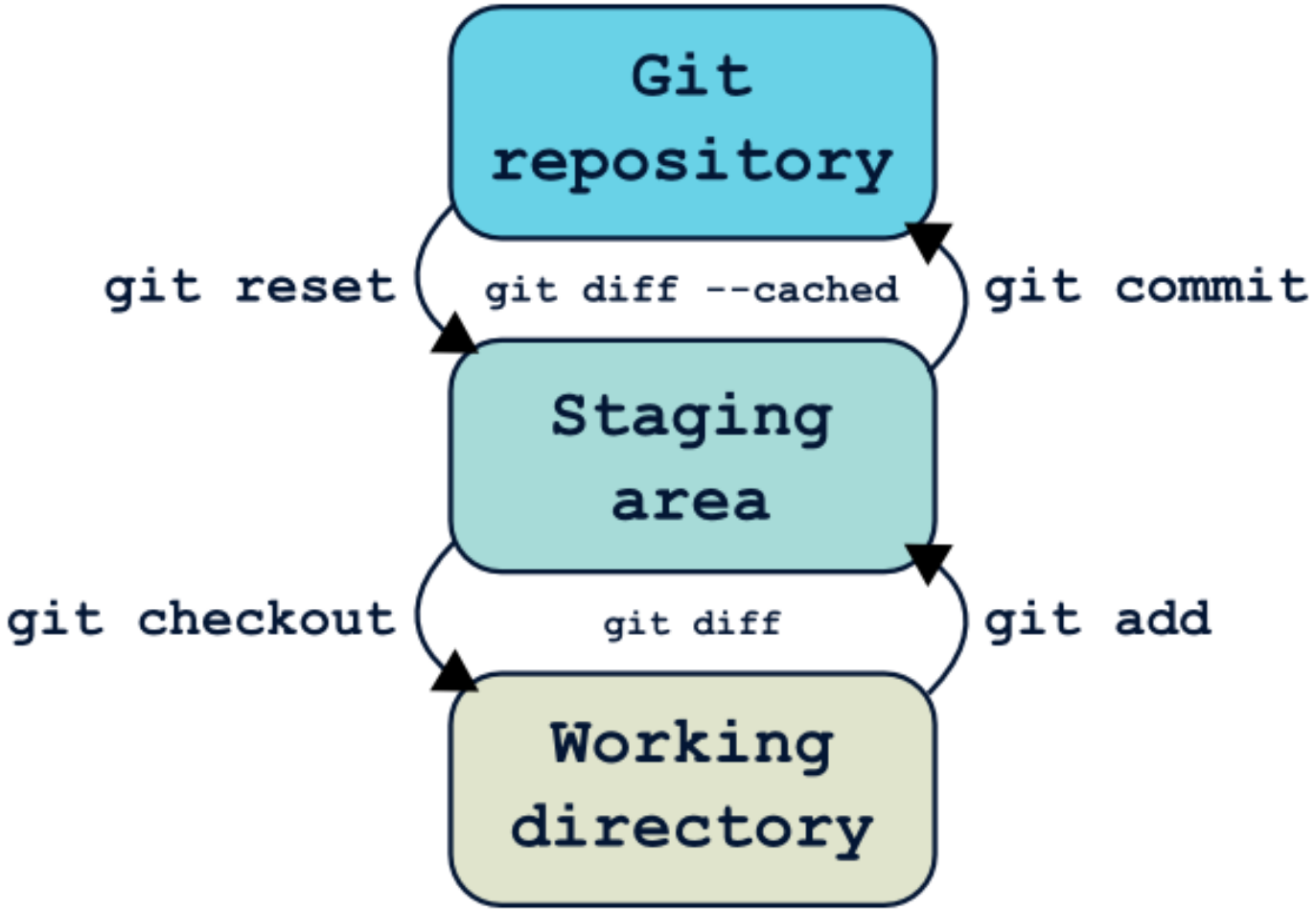
```
mboiziot@Nova ~/workspace/CASSIS $ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   delivery/script/sample/CH3OH_MCMC.py
        modified:   delivery/script/sample/CO_RG.py

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        deleted:    properties/dbHistory.txt
        deleted:    properties/lastFolder.properties
        deleted:    properties/molesRADEXCASSIS.properties

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .pydevproject
        .settings/
        database/template/enable/templatetata.tec
        delivery/config/
        delivery/script/backup/
        properties/bookmark.properties
        properties/jython_last.ini
        properties/openRecent.properties
        properties/sourceSpecies.properties

mboiziot@Nova ~/workspace/CASSIS $
```

Trois états

- Un fichier dans Git dispose de trois états :
  - Working directory : Fichiers dans le répertoire de travail
  - Staging area (ou index) : Fichiers marqués pour le prochain commit
  - Git repository : Fichiers commités



Changer d'états



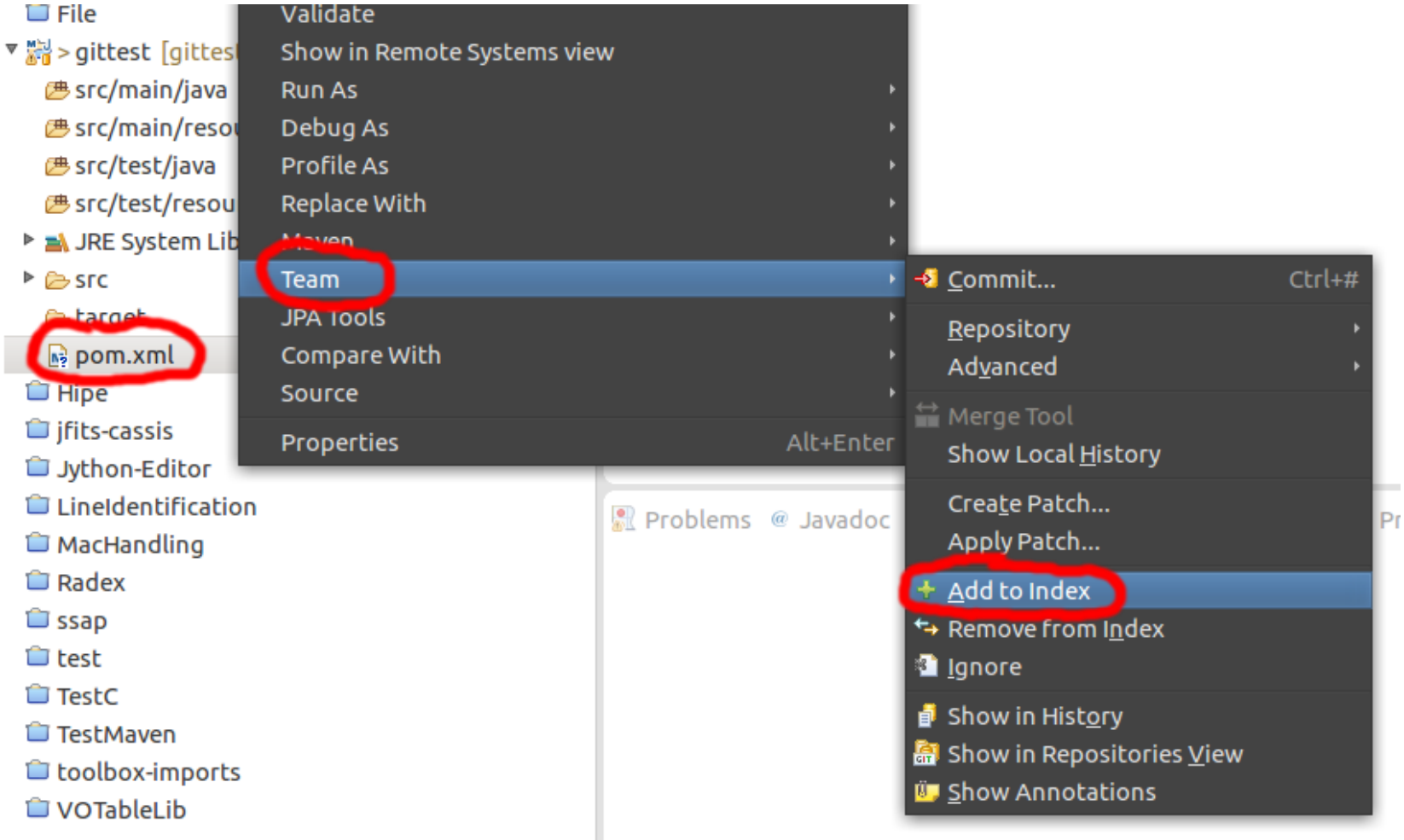
Ajouter des fichiers

- git add <fichiers...>
  - Comprend les expressions régulières
    - git add *.java => Ajout de tous les fichiers Java dans l'index / staging area
    - git add src/ => Ajout de tous les fichiers dans le dossier src
  - Possibilité d'ajout de plusieurs fichiers d'un coup
    - git add src/file1.py src/file2.py

Attention ! Il n'ajoute pas des fichiers mais des modifications !

Si vous faites des modifications entre l'ajout et le commit, elles ne seront pas prises en compte dans le commit.

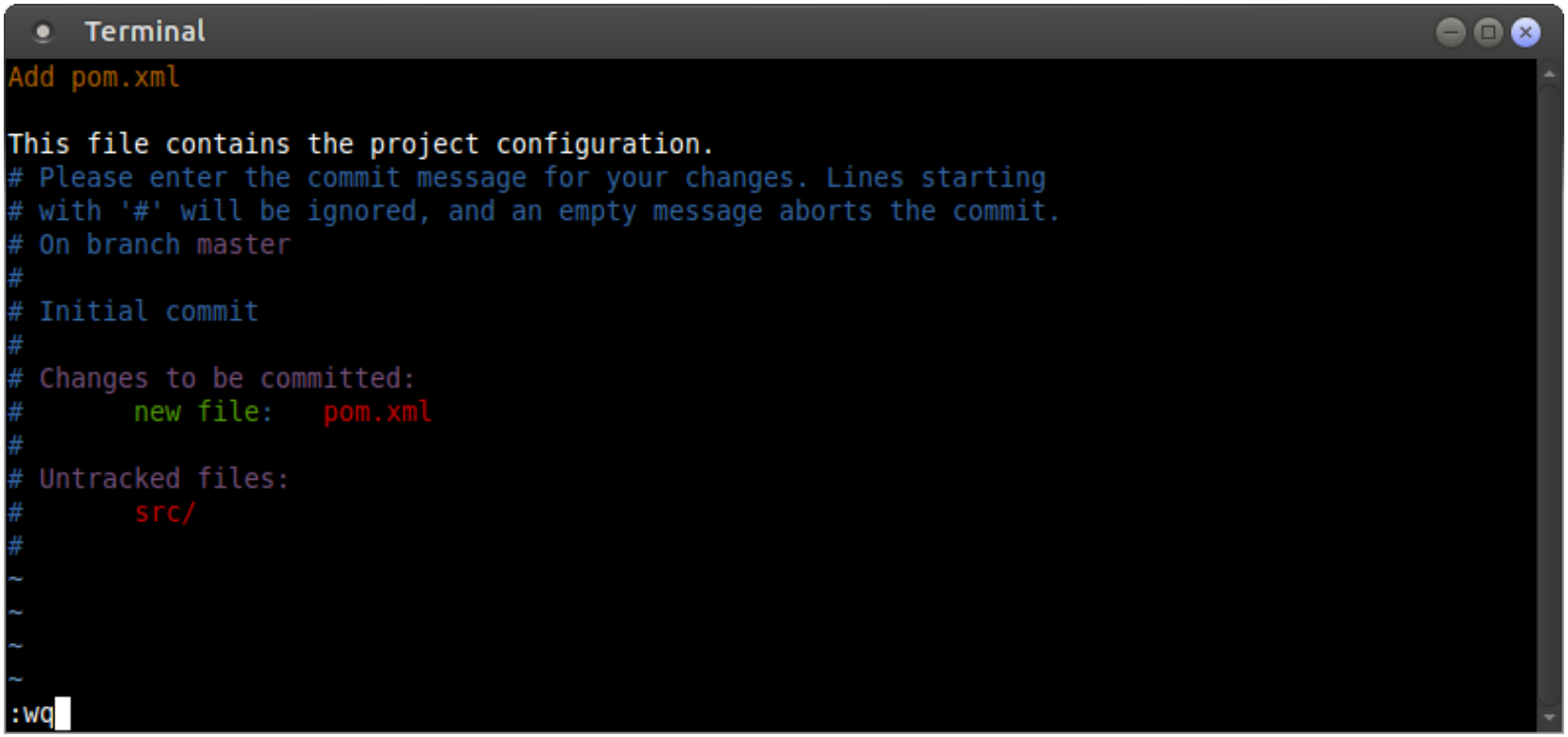
Ajouter des fichiers - Eclipse



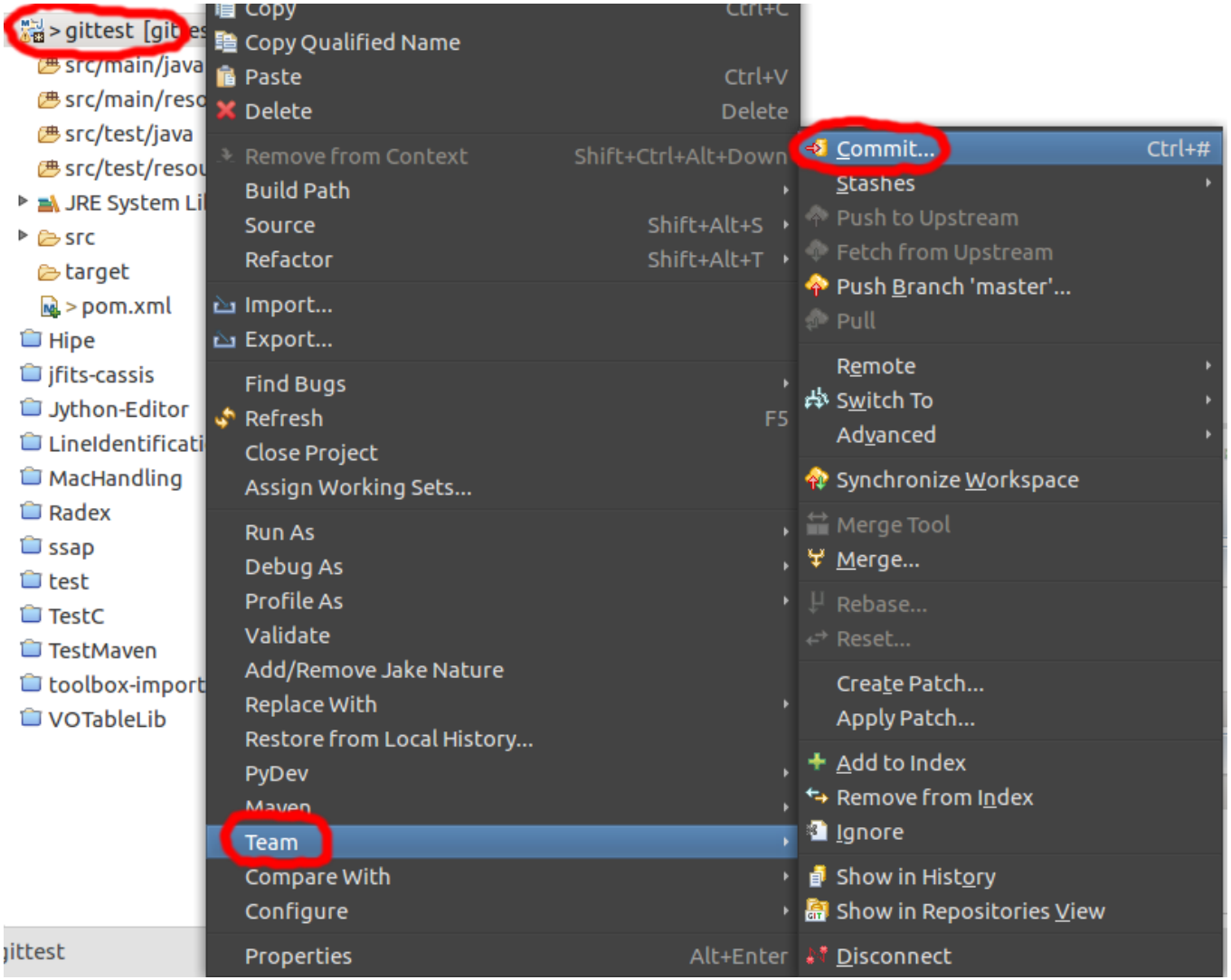
Commiter

- git commit
  - Possibilité d'ajout direct de message de commit via -m "Message"
  - Possibilité d'éditer le commit précédent --amend
  - Fichier commités :
    - Ceux du staging directory si aucun fichier spécifié
    - Possibilité d'en spécifier pour ne commiter que ces derniers
  - Affiche un résumé dans l'éditeur du message de commit
    - Possibilité d'abandon du commit avec un message de commit vide

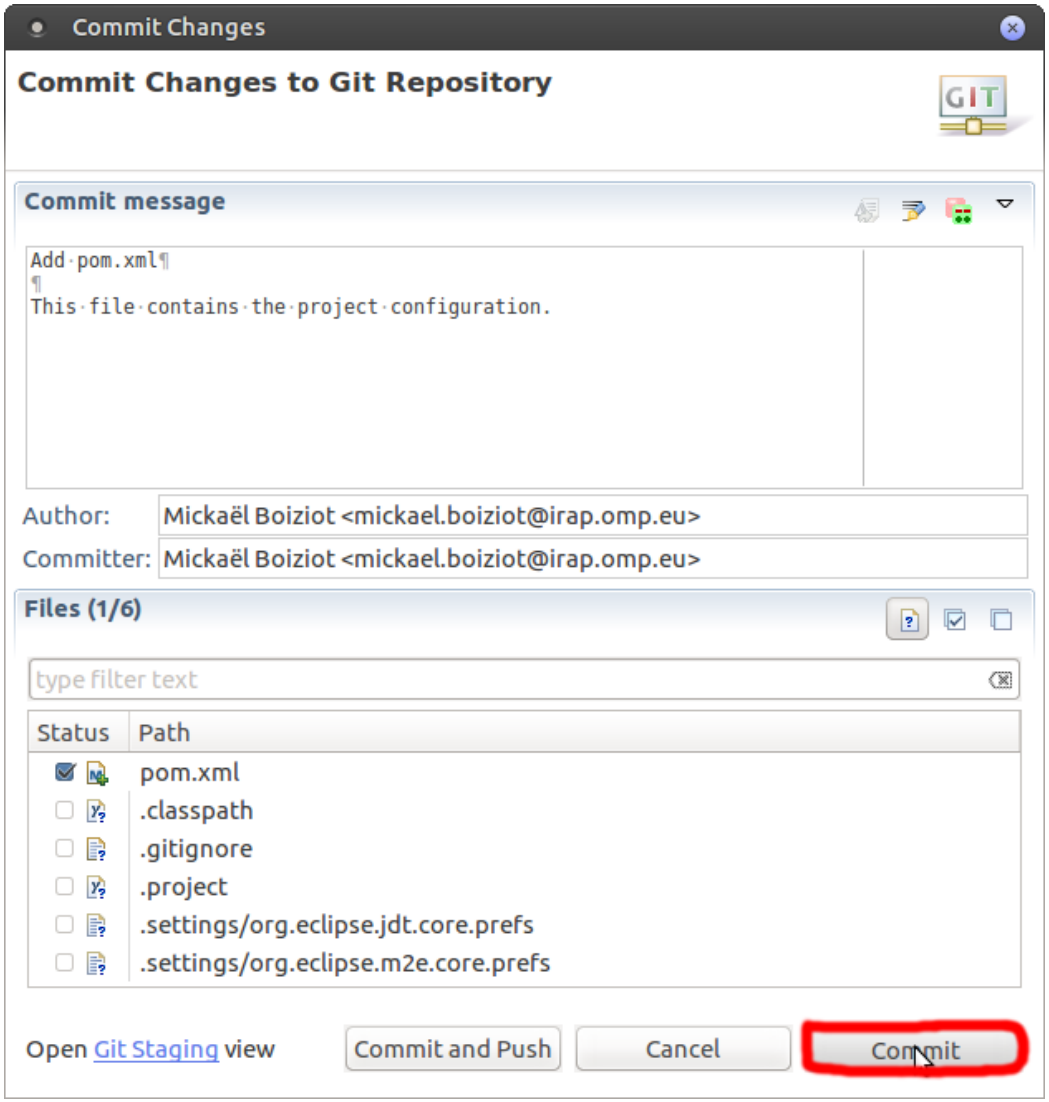
Commiter - exemple



Commiter - exemple avec Eclipse



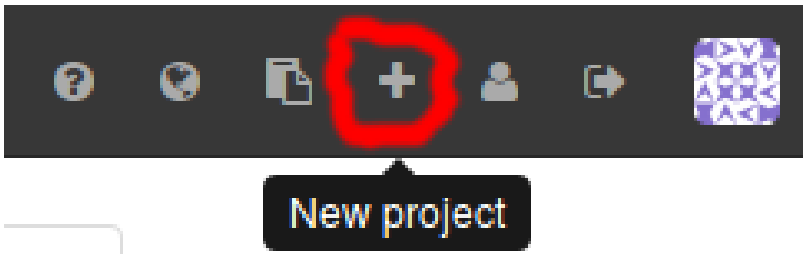
Commiter - exemple avec Eclipse



Connaître l'état : les changements

- git diff
  Informe des changements entre l'index et l'espace de travail
- git diff --cached
  Informe des changements entre le dépot git (HEAD) et l'espace de travail
- git diff <commit1> <commit2>
  Informe des changements entre commit1 et commit2 (possibilité d'inverser l'ordre !)

Création d'un dépot Gitlab



- Remplir le formulaire
  - Possibilité d'utiliser des groupes pour le namespace (collaboratif)

Envoyer sur un dépôt distant

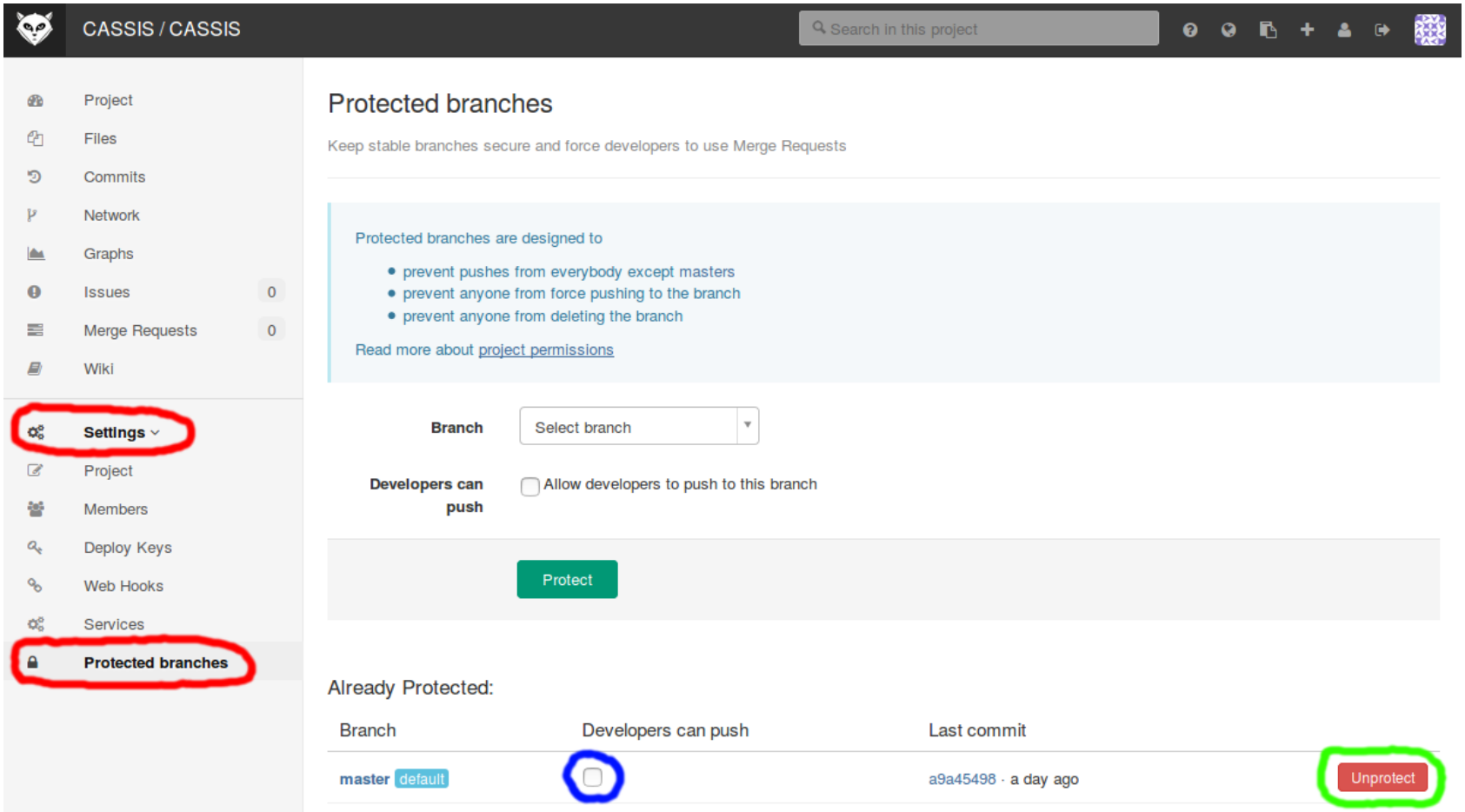
- git push <nom du dépôt> <branche distante>
  - Si dépôt cloné : le nom du dépôt par défaut est "origin"
  - Sinon, il faut l'ajouter avec git remote add <nome du dépot> <adresse>
  - N'envoie par défaut que la branche courante, utilisation de --all pour toutes les envoyer.

Le problème Gitlab

Attention par défaut, impossible pour un développeur de pusher sur la branch master.
Trois solutions :

- Passer l'utilisateur au grade de Master ou Owner sur le projet Gitlab
- Ne plus protéger la branche master (en vert sur le prochain slide)
- Autoriser les développeurs à pusher sur la branche (en bleu sur le prochain slide)

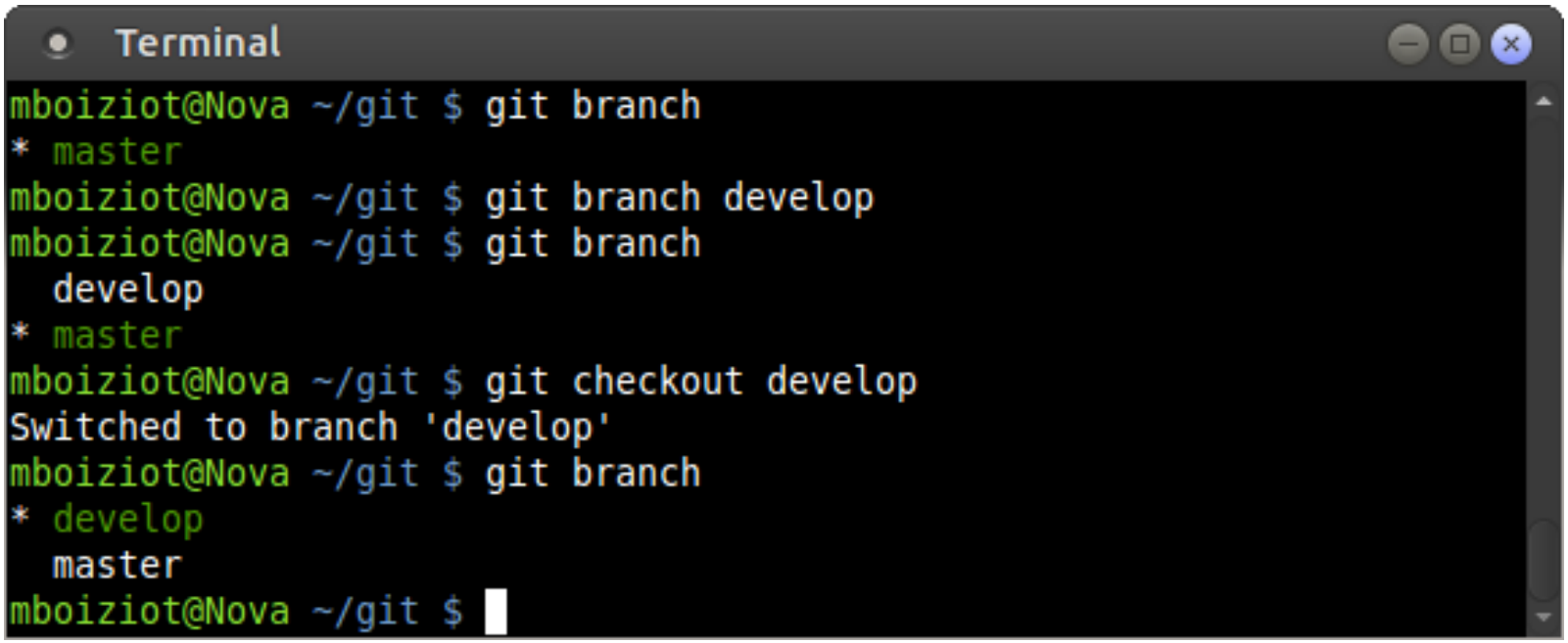
Le problème Gitlab



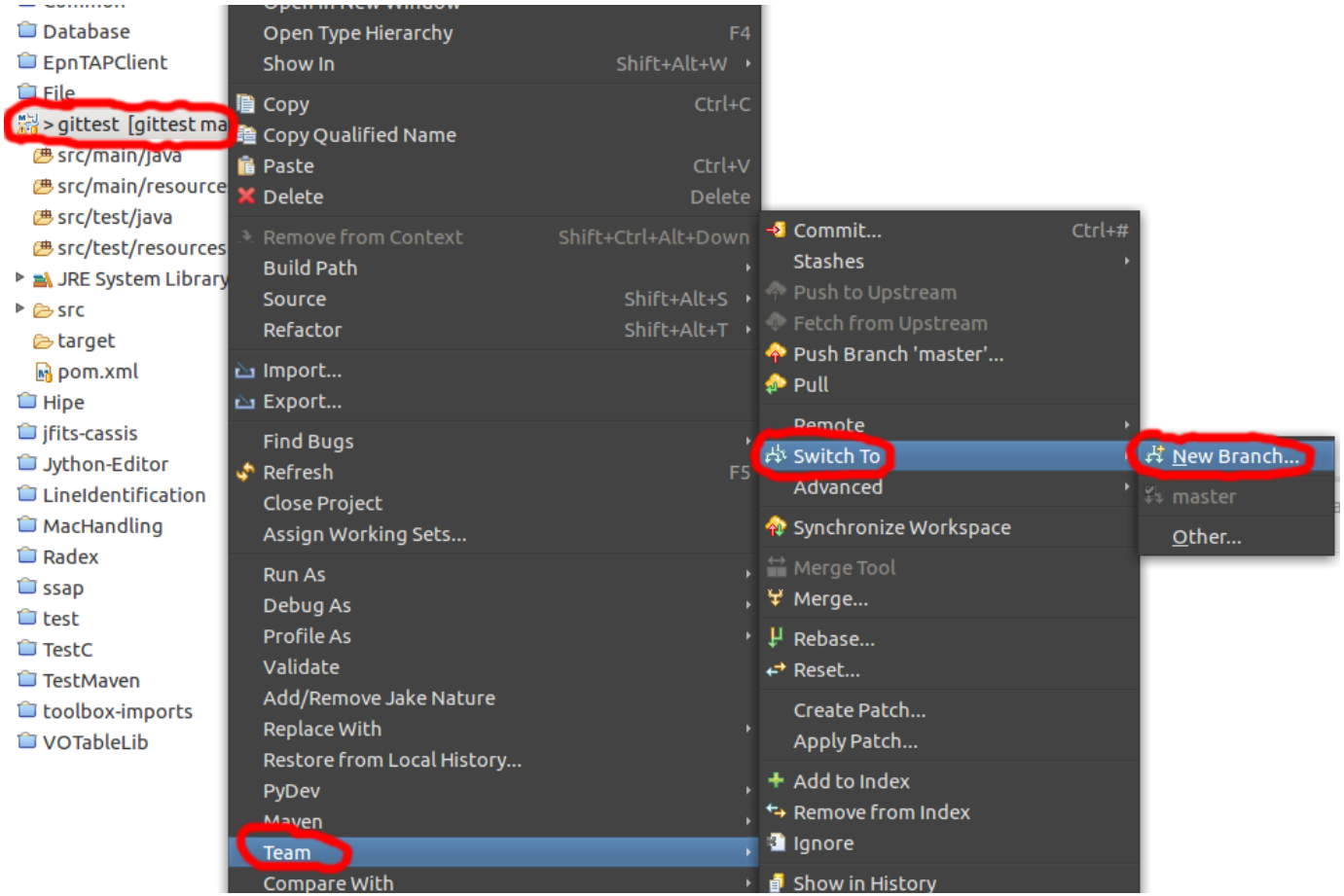
Les branches

- Connaître la branche actuelle : git branch
- Créer une branche : git branch <nom de la branche>
- Changer de branche : git checkout <nom de la branche>
- Supprimer une branche : git branch -d <nom de la branche>

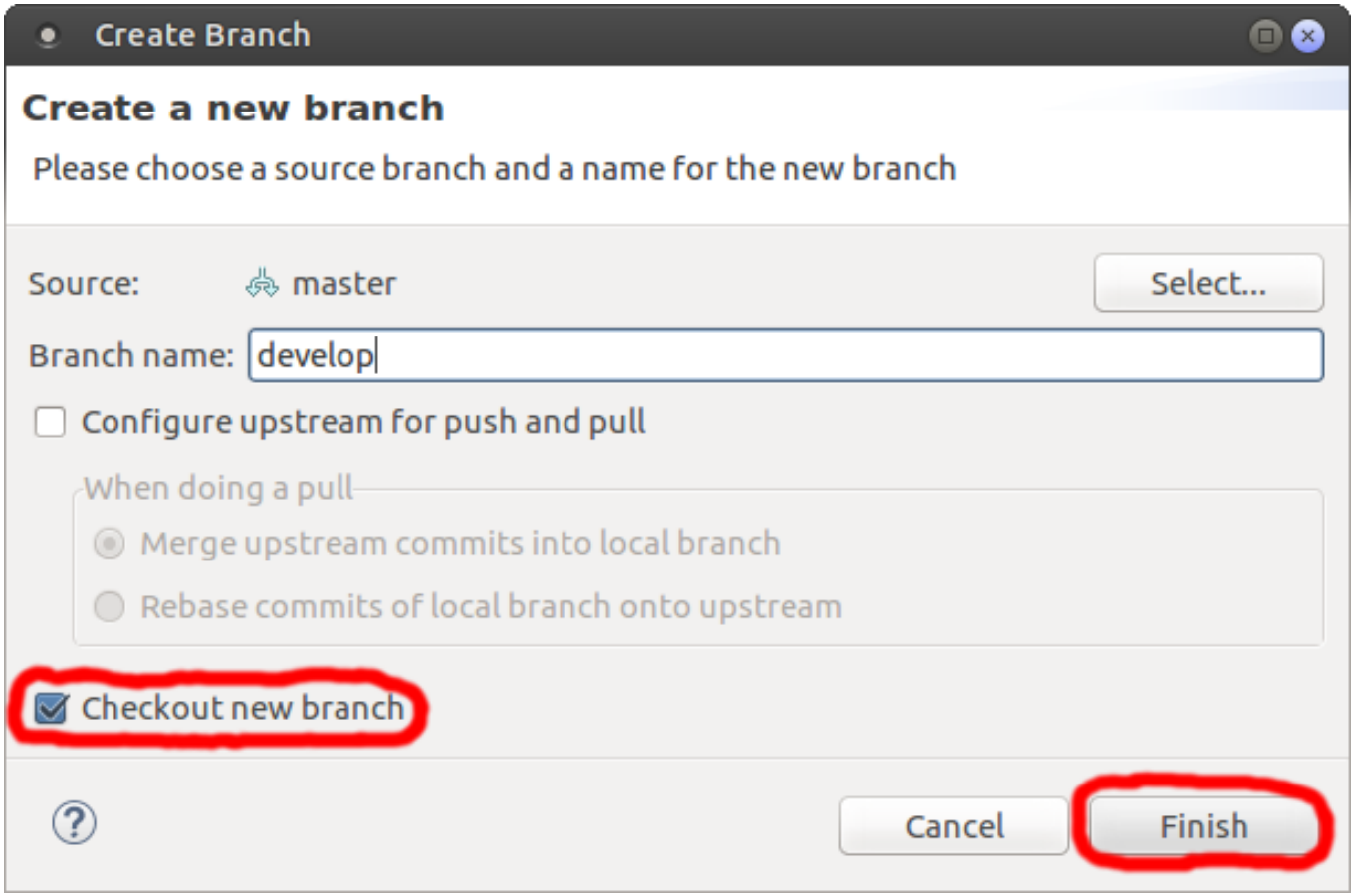
Les branches - exemple



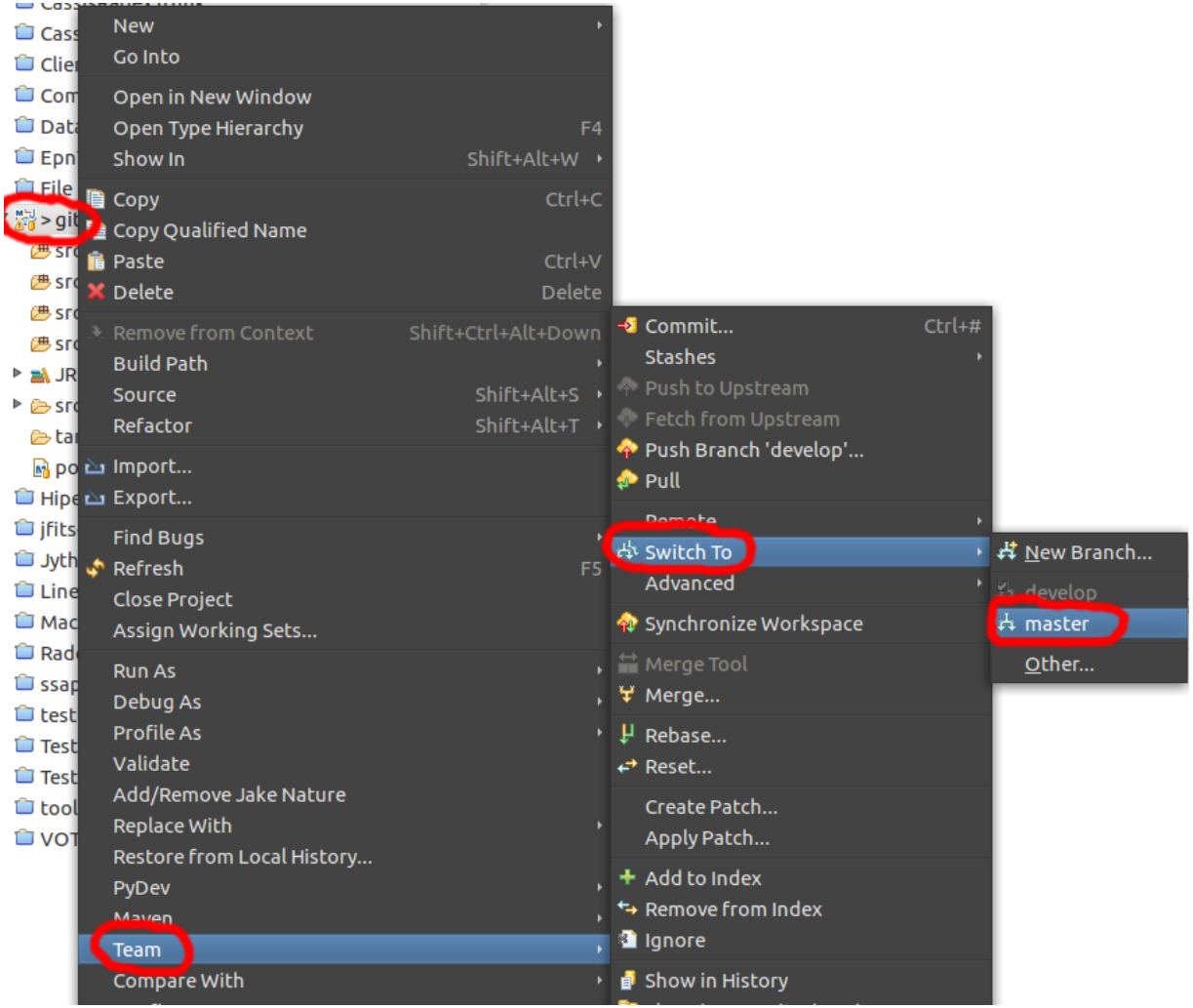
Les branches - exemple avec Eclipse



Les branches - exemple avec Eclipse



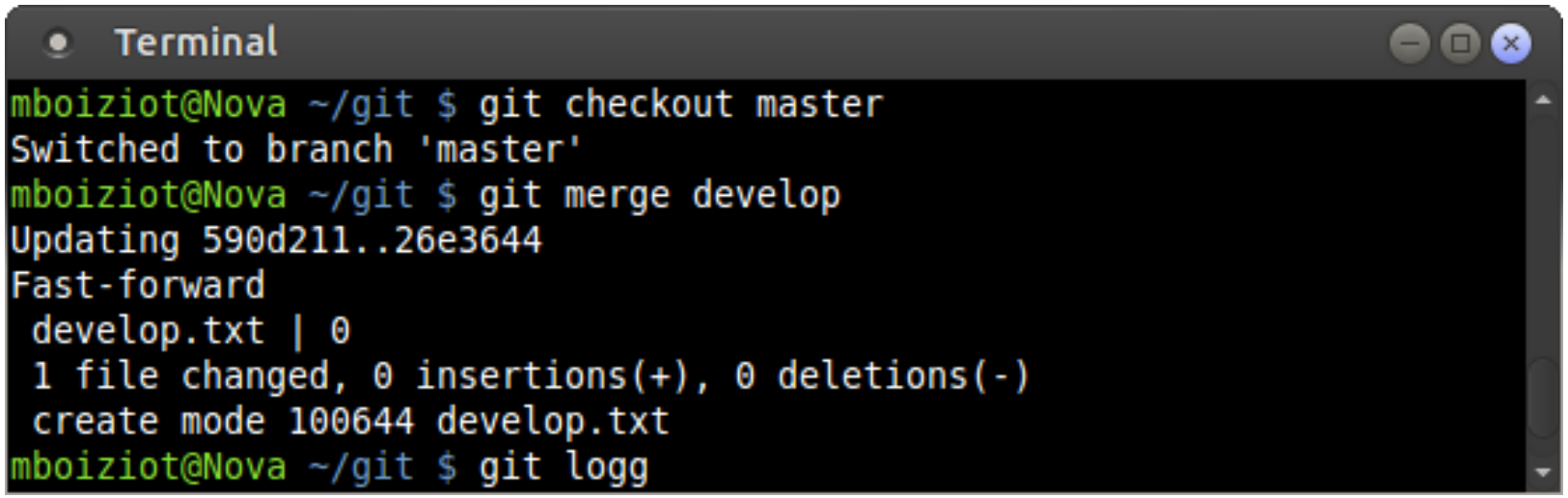
Les branches - exemple avec Eclipse



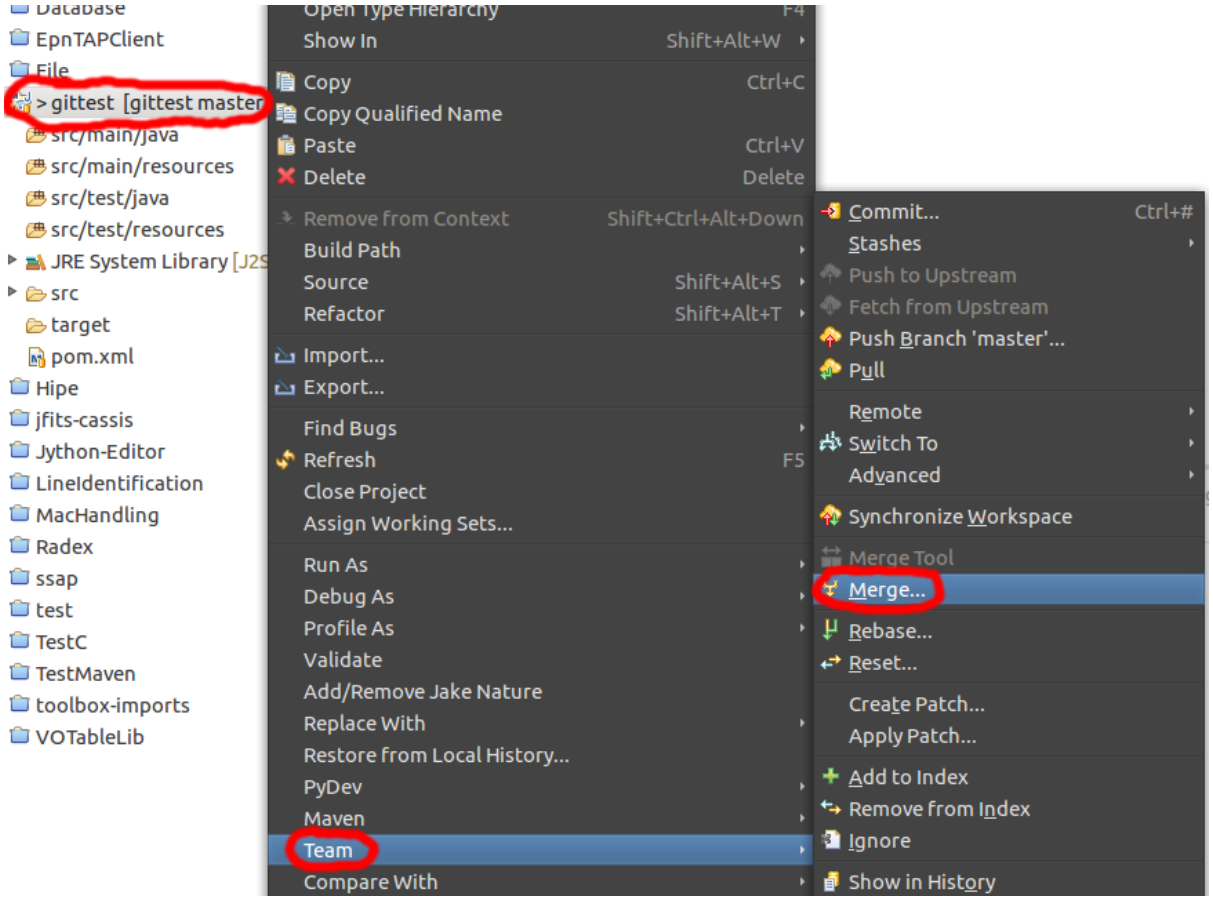
Merger une branche

- Se positionner sur la branche de destination
- git merge <nom de la branche à merger>

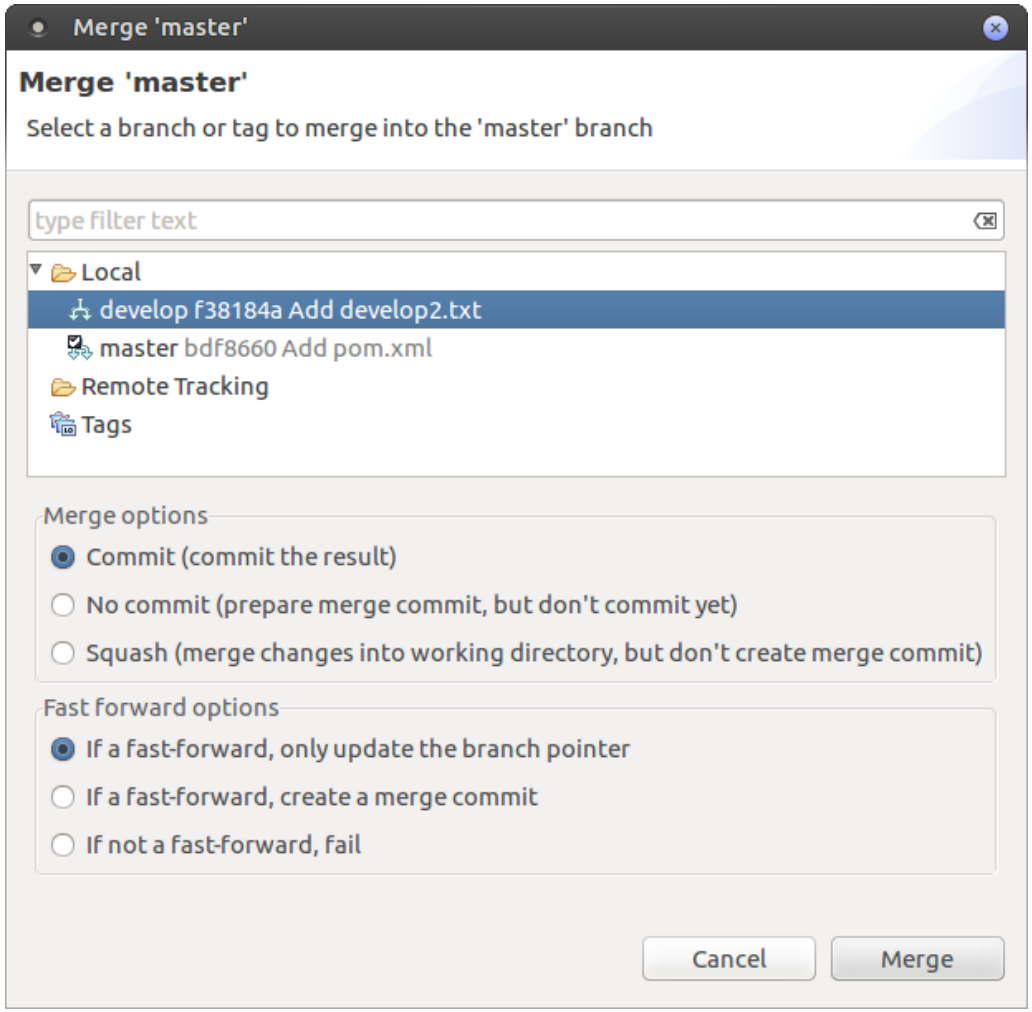
Merger une branche - exemple



Merger une branche - exemple avec Eclipse



Merger une branche - exemple avec Eclipse



Et avec un dépôt existant... ?

- git clone <URL>
- git clone <URL> -b <nom de la branche à merger>
  - Ne télécharge qu'une branche (master ou celle spécifiée)

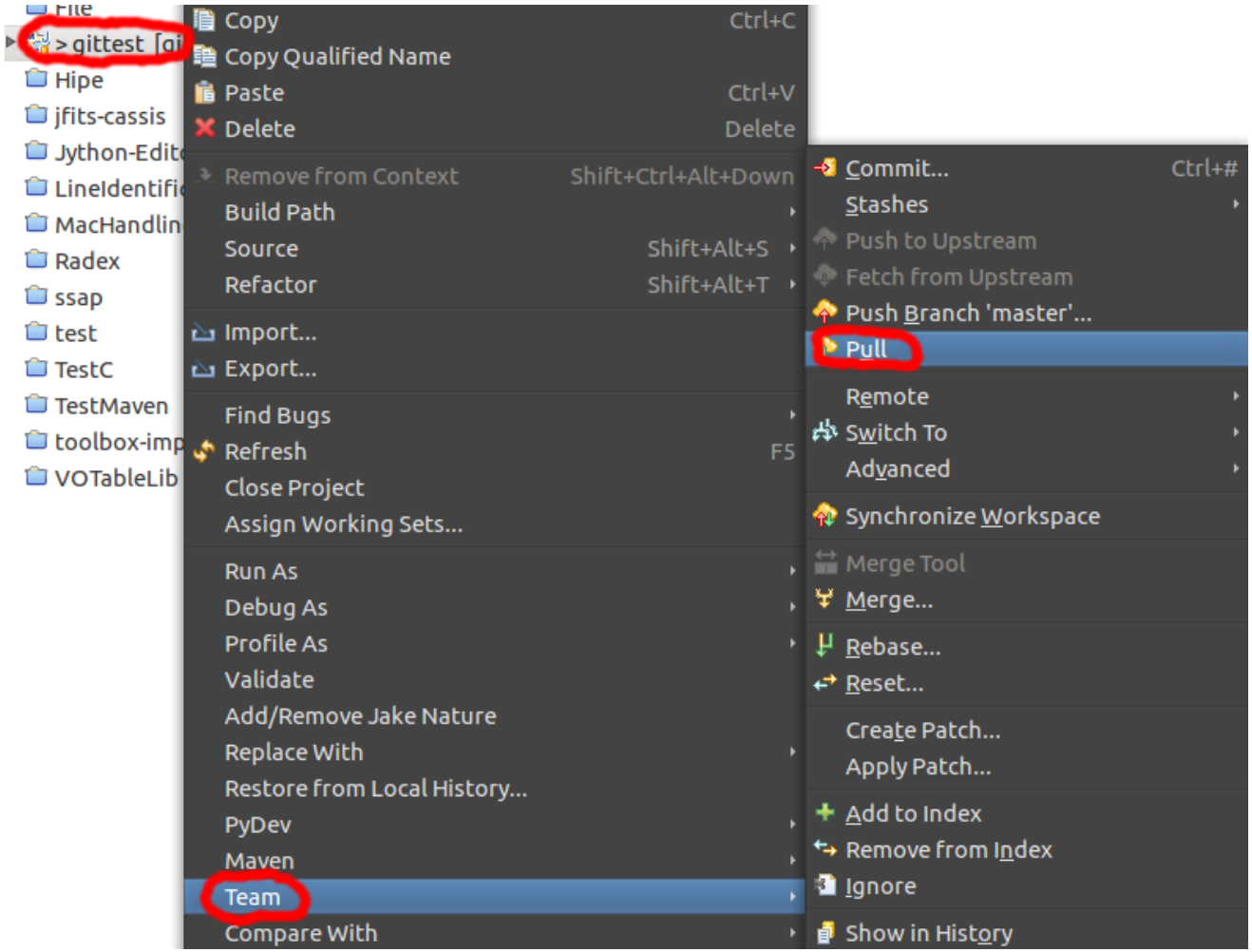
Clone avec Eclipse

- File => Import
- Choisir Git => Projects from Git
- Clone URI
- Copier l'adresse dans URI, Eclipse va compléter le reste automatiquement
- Entrer les identifiants, valider
- Choisir les branches à cloner
- Choisir la destination, nom de dépôt distant
- Import du projet

Récupérer des modifications

- git pull <remote> <branch>
  - Merge toujours local
  - Aucun fichier connu par git ne peut être modifié avant un merge (si c'est le cas, commiter les modifications ou utiliser stash)
  - Equivalent d'un git fetch suivit d'un git merge

Git pull - exemple avec Eclipse



Git tag

Permet de garder un état courant du code.
Exemple : version d'un logiciel.

- git tag <nom du tag>
  - Tag sur HEAD
- git tag <nom du tag> <commit>
  - Tag sur le commit spécifiéq
- Attention par défaut lors d'un push, les tags ne sont pas envoyés. Il faut utiliser l'option --follow-tags

Remote

Permet de gérer les dépôts externes.
Exemple pour avoir un accès via SSH et HTTPS sur le Gitlab !

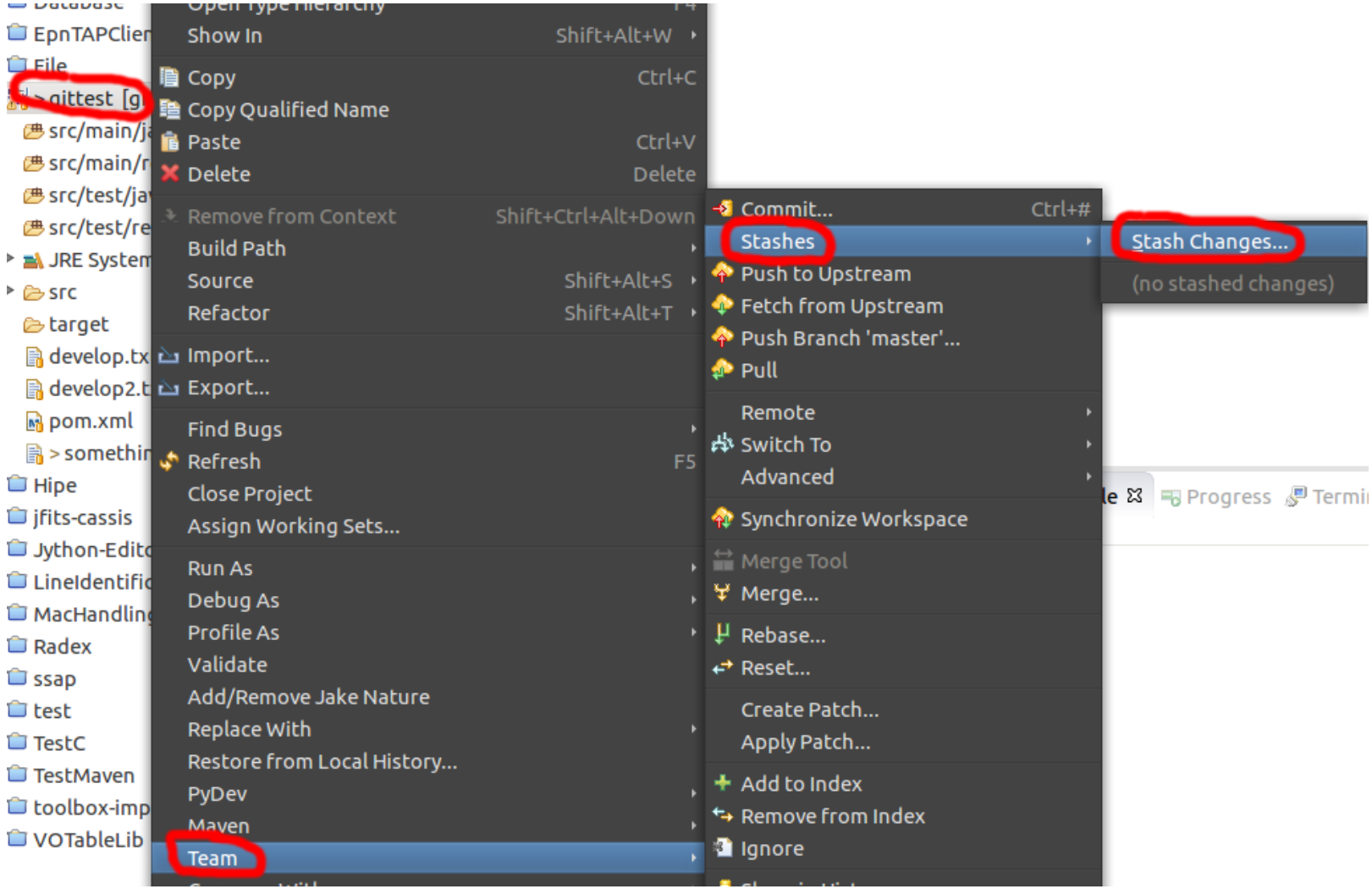
- git remote add <nom> <url>
  - Ajoute un dépôt
- git remote remove <nom>
  - Supprime un dépôt
- git remote rename <ancien nom> <nouveau nom>
  - Renomme un dépôt (utile pour origin !)

Git stash

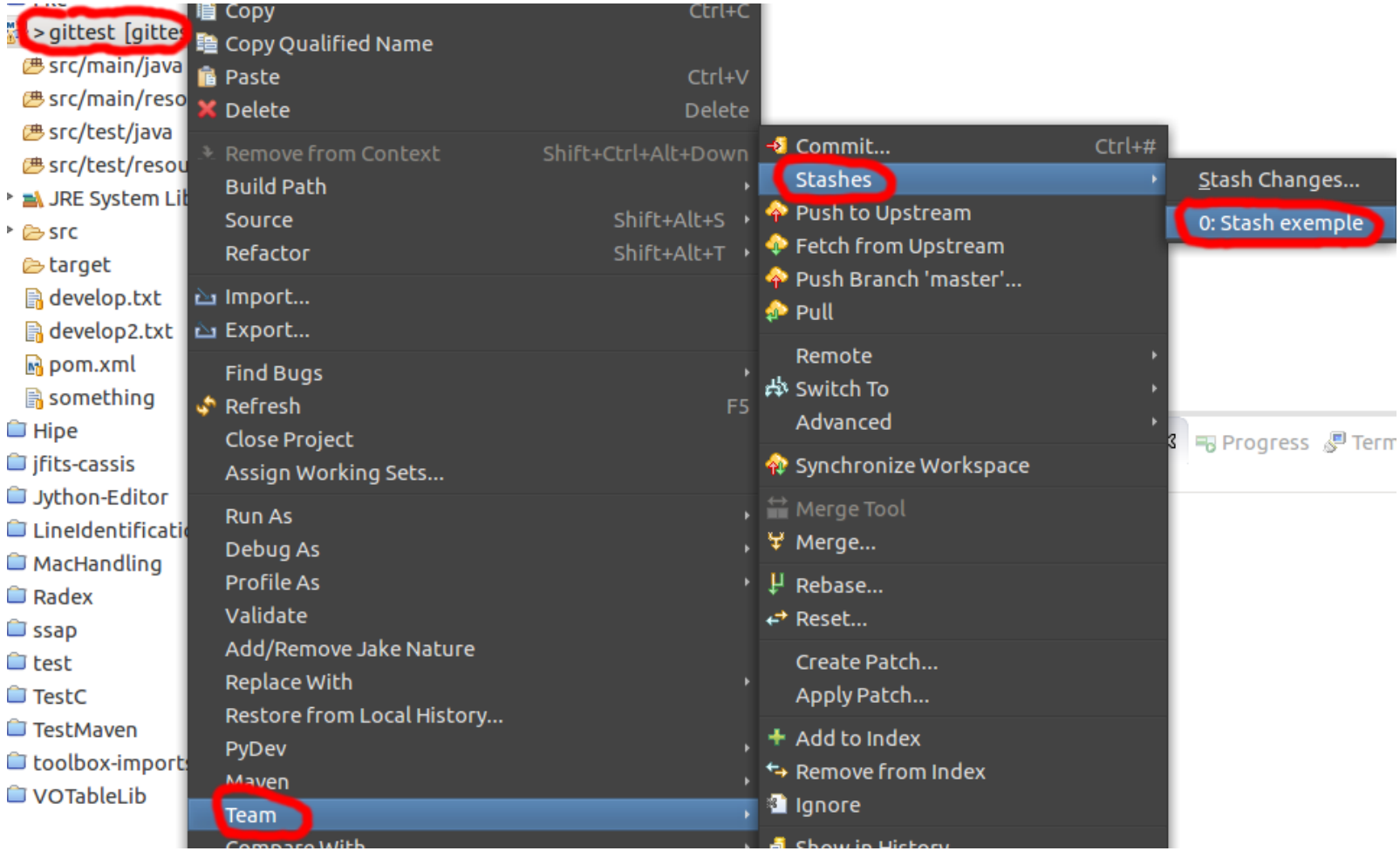
Sauvegarde des modifications locales et nettoie le working directory

- git stash save <message>
  - Sauvegarde des modifications dans un stash
- git stash list
  - Liste les stash
- git stash apply <ref stash>
  - Applique un stash
- git stash pop <ref stash>
  - Applique et supprime un stash

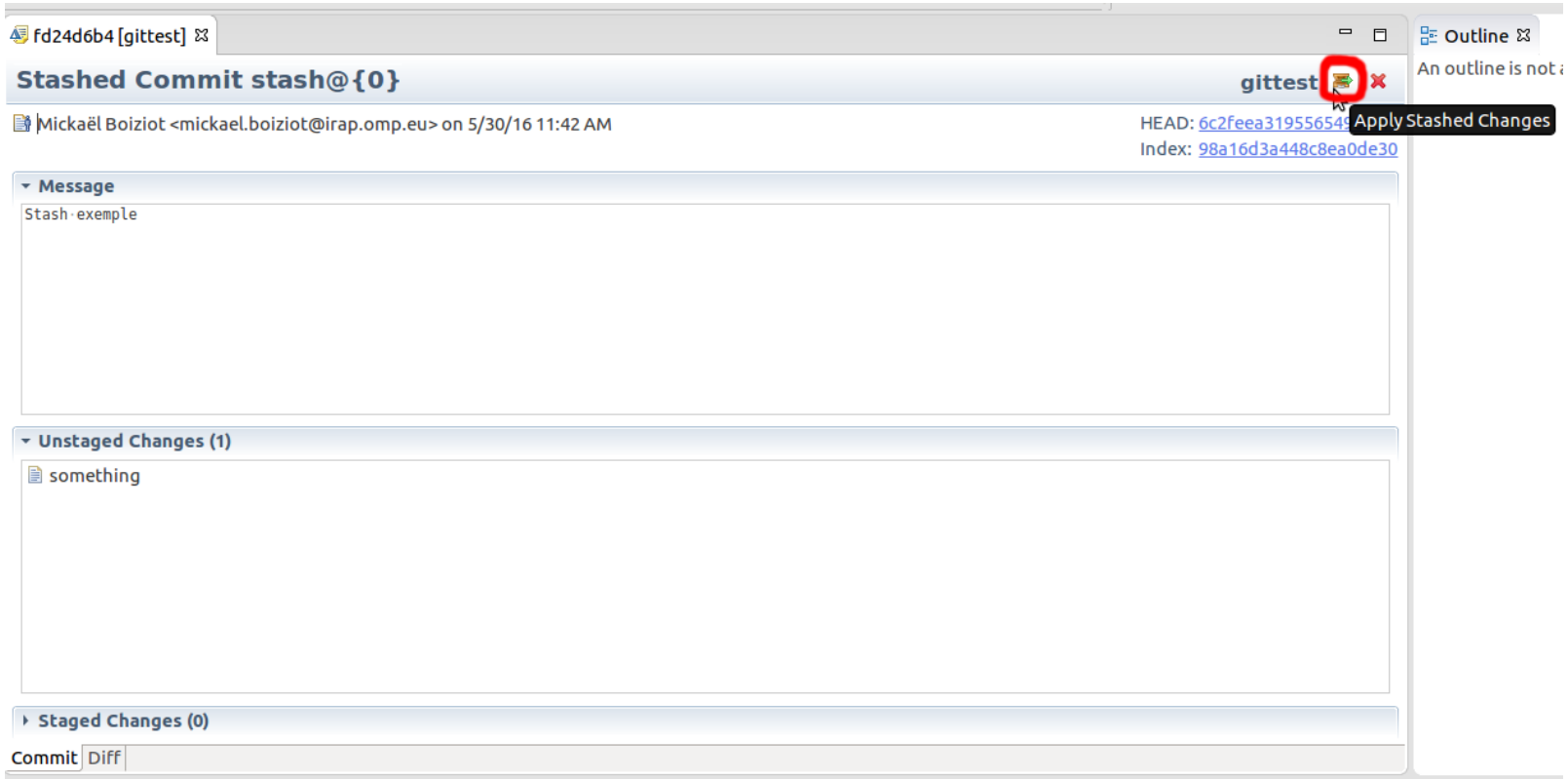
Git stash - exemple avec Eclipse - sauvegarde



Git stash - exemple avec Eclipse - appliquer



Git stash - exemple avec Eclipse - appliquer



Nettoyer le dépôt

Si on veut nettoyer le dépôt local pour le rendre identique.

- git reset --hard
  - Nettoie les fichiers modifiés connus par git
- git clean -f -d -x
  - Supprime les fichiers non connus par git

Gitignore

- Un projet ?
  - source : on veut les commiter
  - fichiers de configuration : on veut les commiter sans les changements de conf local
  - binaire : on ne veut pas les commiter
- La solution : .gitignore, fichier mis par convention dans la racine du dépôt.
  - Contient une liste de fichiers ou expressions régulières
  - N'empêche pas le commit d'un fichier, mais force à l'ajouter au commit explicitement (git add <file>)

Gitignore - exemple

```
# Cassis
doc/
./properties/
Radex/
**/log4j.xml
capture.jpg
myTest.png
*.db-journal
**/CassisVersion.java

# Maven
target/

# Java
*.class
*.jar
hs_err_pid*
```

Gitignore - utilisation avec Eclipse

- Clique droit sur le fichier à ignorer
- Team
- Ignore

Édition - suppression de commit

J'ai commité un mot de passe... je fait quoi ? Deux cas :

- Le commit est sur mon dépôt local et envoyé sur un dépôt partagé (gitlab...)
  - Le service est compromis
  - Je change le mot de passe du service compromis
- Le commit est uniquement sur mon dépôt local
  - Je supprime ou édite le commit
    - Si dernier commit : utilisation de reset (suppression uniquement) : git reset --hard HEAD~1
    - Sinon : utilisation de rebase (suppression, édit...)

Rebase

Permet d'éditer la liste des commits :

- Utiliser le commit
- Editer le message de commit
- Editer le contenu du commit
- Merger deux commits en un seul
- Exécuter une commande sur un commit

Rebase

- git rebase -i HEAD~<nombre de commit à éditer depuis HEAD>
  - Édition d'un fichier pour connaître les opérations à effectuer.
  - Avancement dans les opérations
  - Utiliser commit --amend pour les commit
  - Utiliser git rebase --continue après une opération demandant une intervention.

Liens utiles

- https://git-scm.com/
- https://gitlab.irap.omp.eu/
- http://ndpsoftware.com/git-cheatsheet.html