# Installation de Git sur Redmine

Table des matières

[Installation de Git sur Redmine 1](#_Toc365537311)

[Schéma général de l’organisation Redmine et Git 2](#_Toc365537312)

[Installation et configuration d’un serveur Git (sous Linux) 3](#_Toc365537313)

[Serveur Git pour héberger les dépôts git. 3](#_Toc365537314)

[Création de l’utilisateur git 3](#_Toc365537315)

[Configuration SSH du client 3](#_Toc365537316)

[Configuration sur la machine locale 3](#_Toc365537317)

[Authentification par clé depuis Windows (PuTTY) 3](#_Toc365537318)

[Configuration SSH du serveur 4](#_Toc365537319)

[Création de initialisation d’un dépôt *bare* sur le serveur 5](#_Toc365537320)

[Installation et configuration de Git et d’une GUI sur le poste local 6](#_Toc365537321)

[Téléchargement et installation de msysgit : 6](#_Toc365537322)

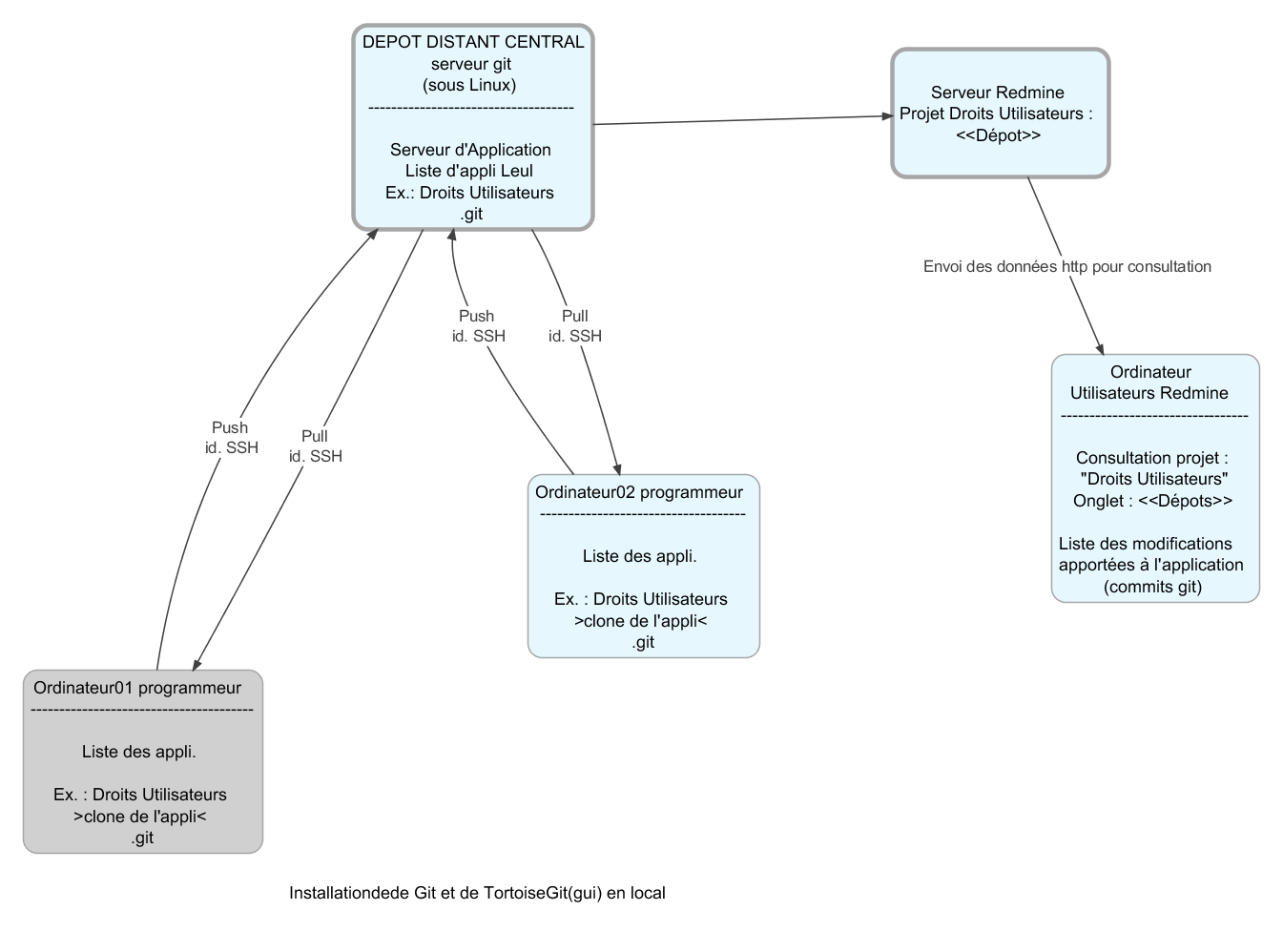
[Téléchargement et installation de Tortoise : 6](#_Toc365537323)

[Lier un dépôt existant à un projet 7](#_Toc365537324)

[Dépôt GIT 7](#_Toc365537325)

[Configuration du projet dans Redmine : 7](#_Toc365537326)

# Schéma général de l’organisation Redmine et Git



L’intégration du SCM Git dans notre environnement de programmation ne devrait pas fondamentalement changer nos habitudes.

Sur le serveur d’applications nous aurons toujours nos versions de programmes testés et exploitables.

Sur les postes des programmeurs, nous aurons les applications en cours de mise à jour.

Les fichiers changelog déposé actuellement avec chacun de nos programmes sur le serveur d’applications corresponde aux futurs commits de git (récapitulatif des modifications effectuées pour la nouvelle version en cours).

L’avantage d’intégrer ce nouveau SCM dans notre environnement de programmation et de l’intégrer dans Redmine nous permettra d’automatiser et de sécuriser le rapatriement sur le poste du programmeur et la mise à jour sur le serveur d’applications, des programmes à modifier. On pourra aussi si nécessaire revenir sur des versions précédentes. On pourra aussi nommer les mises à jour importantes par des tags.

Le deuxième avantage par rapport à notre environnement actuel, sera de pouvoir suivre en temps réel à travers Redmine, les modifications qui sont appliquées à chacune de nos applications. On pourra connaître quand, par qui et quoi a été fait lors des dernières modifications.

# Installation et configuration d’un serveur Git (sous Linux)

## Serveur Git pour héberger les dépôts git.

Sur Linux, Git est disponible dans le gestionnaires de paquets.

sudo apt-get install git

## Création de l’utilisateur git

Un utilisateur git permet de connecter sur la machine en SSH.

sudo adduser --system --shell /bin/bash --group --disabled-password --home /var/git/

Changer le propriétaire du répertoire « /var/git/ » pour l’utilisateur « git »

sudo chown git :git /var/git

## Configuration SSH du client

Pour accéder à partir de machines distances, on utilise SSH pour se connecter en tant qu’utilisateur git.

### Configuration sur la machine locale

Création d’une clé SSH :

**Sous linux :**

ssh-keygen –t rsa

**Sous Windows :**

Puttygen

### Authentification par clé depuis Windows (PuTTY)

Les clés permettent de s’authentifier sur le serveur sans avoir à saisir d’identifiant.

Le client génère une clé publique et une clé privée.

Clé privée  : « id\_rsa »

Clé publique  : « id\_rsa.pub »

Envoie des clés sur le serveur.

* Génération des clés

1 – Lancer PuttyGen

2 – Cliquer sur le bouton « Generate » (option : SSH-2 RSA 1204)

3 – Saisir une passphrase (zones « Key passphrase » et « Confirm passphrase »)

4 – Enregistrer la clé publique – Cliquer « Save public key » (nom et emplacement quelconques)

5 – Enregistrer la clé privée – Cliquer « Save private key » (extension .ppk)

Créer un dossier .ssh dans votre Home (Sur Windows C:/Utilisateurs/NomUtilisateur/) et y enregistrer la clé privé.

## Configuration SSH du serveur

Fin de la configuration SSH sur le serveur :

sudo mkdir /var/git/.ssh

sudo touch /var/git/.ssh/authorized\_keys

Écriture du contenu de la clé publique dans le fichier « authorized\_keys »

echo ‘’LE\_CONTENU\_DE\_LA\_CLE’’ >> /var/git/.ssh/authorized\_keys

Vérification des droits des fichiers de configuration SSH :

chmod 755 ~/.ssh

chmod 644 ~/.ssh/authorized\_keys

## Création de initialisation d’un dépôt *bare* sur le serveur

Un dépôt bare est un dépôt vide prêt à recevoir un nouveau projet.

sudo mkdir /var/git/NomDuProjet.git

sudo cd/var/git/NomDuProjet.git

sudo git init --bare

sudo chown -R git:git /var/git/NomDuProjet.git

# Installation et configuration de Git et d’une GUI sur le poste local

## Téléchargement et installation de msysgit :

Msysgit est le programme git sans interface graphique. Il faut utiliser les lignes de commandes dans la console Git bash.

<http://code.google.com/p/msysgit/downloads/list?q=full+installer+official+git>

Prendre la dernière version.

* Etapes de l’installation :

1 – Sélectionner un dossier d’installation

2 – Décocher l’option : Windows Explorer intégration

3 – Sélectionner le dossier du menu de démarrage

4 – Use Git Bash only

5 – Checkout Windows-style, commit Unix-style line endings

## Téléchargement et installation de Tortoise :

Tortoise est l’interface graphique pour une utilisation plus naturelle de git.

Il existe d'autres interfaces graphiques pour git.

<http://code.google.com/p/tortoisegit/wiki/Download>

Choisir la bonne version de plate-forme.

* Etapes de l’installation :

1 – Sélectionner un dossier d’installation

Installation de la langue française :

<http://code.google.com/p/tortoisegit/wiki/Download>

Choisir la même plate-forme que le programme.

* Etapes de l’installation :

1 – Exécution du fichier d’installation.

2 – Menu « Démarrer / Tous les programmes / TortoiseGit / Settings »

3 – Section « Général » - Language : choisir français.

4 – Cliquer « Ok »

# Lier un dépôt existant à un projet

## Dépôt GIT

Le dépôt git *bare* a été créé à la section « [Création de initialisation d’un dépôt *bare* sur le serveur](#_Création_de_initialisation)»

## Configuration du projet dans Redmine :

Dans redmine, vérifier que le module 'Dépôt' est activé :

1 – Aller sur l'onglet 'Dépôt'.

2 – Choisir Git comme SCM.

3 – Entrer le chemin ou l'URL du dépôt. Ex. : /var/git/NomDuProjet.git

**Important** :

Lorsque vous parcourez le dépôt pour la première fois, Redmine récupère la description de tous les commits (nos maj) existants et les stocke en base de données. Cette opération est réalisée en une seule fois par dépôt, mais peut s'avérer très longue (allant même jusqu'au time out) si votre dépôt a des centaines de commits.

Pour éviter cela, vous pouvez le faire hors ligne.

Après avoir déclaré votre dépôt dans Redmine, lancez la commande suivante :

ruby script/runner "Repository.fetch\_changesets" -e production

Tous les commits seront récupérés dans la base de données de Redmine.

Depuis Redmine 0.9.x, on peut utiliser un lien pour exécuter « fetch\_changesets » pour un projet spécifique ou sur la totalité.

* Récupérer les changements pour tous les projets actifs :

<http://redmine.example.com/sys/fetch_changesets>

* Récupère les changements pour le projet foo seulement :

<http://redmine.example.com/sys/fetch_changesets?id=foo>