TP2 - Déploiement d'un serveur Git

Aujourd'hui, Git est de loin le système de contrôle de version le plus largement utilisé. C'est un projet open source avancé, qui est activement maintenu. Apparu en 2005, c'est Linus Torvalds, le créateur bien connu du noyau du système d'exploitation Linux, qui l'a développé au départ. De plus en plus de projets logiciels reposent sur Git pour le contrôle de version, y compris des projets commerciaux et en open source. L'objectif de ce second TP est double :

- *installer un serveur Git en local sur la machine ;*
- *y accéder depuis un autre ordinateur client via une clé ssh.*

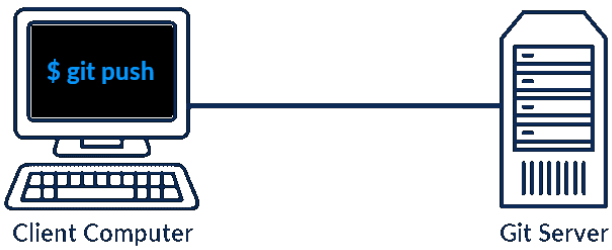
Ce TP s'appuie sur un tutoriel de Brandon Wallace, développeur d'applications web. Le tutoriel est intitulé ***Set Up A GIT Server The Easy Way !****.*

Comme d'habitude, il peut être nécessaire d'être connecté en super-utilisateur (`root`). Seules les grandes lignes des commandes seront décrites, pour avoir la syntaxe complète d'une commande on vous invite à utiliser le manuel : `man [commande]` (`[ ]` indique que c'est optionnel), exemple : `man ls`.

IMPORTANT : changer le mot de passe de root ! !

1 Introduction

Un serveur Git peut être déployé / installé sur un ordinateur faisant office de serveur, mais également sur des plateformes plus légères telles qu'un BeagleBone Black ou un Raspberry Pi, ou encore une machine virtuelle. Comme indiqué précédemment, nous allons installer `git` sur la plateforme faisant office de serveur, à savoir votre machine, et utiliser une clé `ssh` sur le client pour y accéder, sachant que le client sera un de vos voisins. À la fin du TP vous aurez votre propre serveur Git privé, qui vous permettra de cloner un dépôt, de pousser vers un dépôt des modifications locales, etc.



C'est parti...

2 Sur la machine serveur

Pour commencer, il s'agit d'installer `git` après vous être connecté en `root`. Puis de créer et configurer le pseudo-utilisateur *git*.

Les manipulations à effectuer sont :

1. installation de `git`

   ```
   su -
   apt update
   apt upgrade
   apt install git
   ```

2. création de l'utilisateur *git* avec la commande `useradd`

   ```
   useradd -r -m -U -s /bin/bash git
   ```

   Utiliser le manuel pour voir le rôle de chacune des options de la commande. Vérifier que le `home directory` de l'utilisateur *git* est bien créé via `ls -lF /home/` ;

3. définir un mot de passe pour l'utilisateur *git* ;
4. se connecter avec `su` en tant qu'utilisateur *git* ;
5. créer un répertoire `.ssh` dans le *home directory* de *git* et modifier les droits du répertoire pour qu'ils soient

   ```
   drwx------
   ```

   Vérifier que vous obtenez le résultat attendu avec `ls -ld /home/git/.ssh` ;

Passons maintenant à la création du premier dépôt sur le serveur. Par convention on ajoute `.git` à la fin du répertoire correspondant, cela nous permettra de rappeler qu'il s'agit d'un répertoire de type *repository* et non pas un répertoire classique.

6. on utilise la commande `git`

   ```
   git init --bare my_project.git
   ```

C'est fini pour le serveur !

3 Sur la machine cliente

ATTENTION : il s'agit de faire les manipulations vous permettant d'accéder au serveur Git d'un(e) de vos voisin(e)s. Il faut donc en particulier adapter l'adresse IP utilisée ci-dessous par rapport à celle de la machine du / de la voisin(e).

Dans un premier temps il s'agit de créer une paire de clé RSA, puis de copier la clé publique dans le compte *git* du serveur distant. Cela doit être fait dans le compte de l'utilisateur `tpreseau`. Vous vérifierez bien ce point avant de commencer.

Les manipulations à effectuer sont :

1. générer une paire de clés RSA de 4096 bits, en faisant en sorte que le nom des fichiers soit `id_rsa_git`. Pour cela vous utiliserez les options `-t`, `-b` et `-f` ;
2. vérifier que la paire de clés est bien générée via `ls -la .ssh` ;
3. copier la clé publique avec la commande `ssh-copy-id` dans le compte de l'utilisateur *git* sur la machine hébergeant le serveur `git`.
4. vérifier que vous arrivez à vous connecter dans le compte *git* en utilisant la paire de clés. Après connexion, regarder le contenu du fichier `authorized_keys` qui se trouve dans le répertoire `.ssh` de l'utilisateur *git* ;

5. finalement se déconnecter du serveur.

La seconde étape consiste à créer un répertoire local, à l'initialiser et ajouter quelques fichiers, enfin à le lier avec le dépôt créé précédemment sur le serveur distant et qui enregistrera les commits.

Les manipulations à effectuer sont :

1. créer dans le *home directory* du compte *tpreseau* de votre machine un répertoire avec le même nom que le premier dépôt sur la machine serveur, mais sans l'extension `.git` :

   ```
   mkdir my_project
   ```

2. descendre dans le répertoire et y créer quelques fichiers Git "standards"

   ```
   cd my_project
   touch readme.md .gitignore LICENSE
   ```

   on ajoute un commentaire dans le fichier `readme.md`

   ```
   echo '# Mon Premier Projet!' >> readme.md
   ```

3. création de la Git *staging area* qui va servir de zone intermédiaire entre un répertoire local et un dépôt (git, github, ...) sur un serveur

   ```
   git init .
   ```

4. on ajoute les fichiers, on commit, puis on regarde le résultat

   ```
   git add --all
   git commit -m "Premier commit"
   git log
   git status
   ```

5. maintenant on définit le serveur Git qui sera utilisé pour enregistrer les commits. Dans la commande on donne le nom suivant au serveur distant : `git_X`, avec `X` qui est le numéro de la machine où se trouve le repository. La syntaxe de la commande est :
   `git remote add <remote_name> git@<server_ip>:<git_repo>`.
   Dans notre cas cela donne (on rappelle qu'il faut adapter l'adresse IP) :

   ```
   git remote add git_X git@192.168.56.201:my_project.git
   ```

6. on pousse le premier commit sur le serveur

   ```
   git push git_X master
   ```

   Si c'est le mot de passe qui est demandé, c'est que l'authentification par clé ne se fait pas de manière automatique, auquel cas arrêter la manipulation et résoudre le problème (voir ci-après). Une première solution, non pérenne, consiste à utiliser la commande `ssh-add` pour mémoriser la clé. Une seconde solution est d'ajouter (ou créer) une entrée dans le fichier `config` du répertoire `.ssh` de *tpreseau* comme suit :

   ```
   Host gitserver
       Hostname 192.168.56.201
       IdentityFile ~/.ssh/id_rsa_git.pub
       IdentitiesOnly yes
   ```

   auquel cas la commande `git remote add` deviendrait

   ```
   git remote add git_X git@gitserver:my_project.git
   ```

Le serveur étant fonctionnel comme vous avez pu le constater avec la mise en place du premier dépôt, nous allons voir comment créer directement un nouveau dépôt sur le serveur sans avoir besoin de se "connecter" à celui-ci. En fait, il s'agit d'utiliser SSH pour exécuter des commandes à distance sans maintenir une connexion ouverte en permanence.

Les manipulations à effectuer sont :

1. créer le deuxième dépôt en utilisant la même commande que pour le premier, mais en l'appelant `my_second_project` et en utilisant `ssh`. Vérifier avec un `ls`, toujours exécuté à distance avec `ssh` que les deux dépôts sont bien présents dans le compte de `git`.
2. sur la machine cliente on crée un répertoire local comme précédemment, mais avec comme commentaire (dans `readme.md`) `# Mon Deuxieme Projet!` ;
3. faire les manipulations adéquates au niveau de Git et pousser le projet.

Finalement, vous pourriez automatiser tout cela en créant un script qui prend comme paramètre le nom du répertoire local et enchaîne les commandes ci-dessus...

Les manipulations à effectuer sont :
écrire un tel script.

Pour ceux qui arriveraient là et auraient encore du temps, regarder comment il est possible de modifier la configuration du serveur `ssh` (le manuel est votre ami...).

Paramétrage du serveur SSH

Dans le fichier définissant la configuration du serveur (ou démon) `ssh`, toutes les valeurs commentées (précédées d'un #) sont placées à leur valeur par défaut. Par exemple, la ligne `#PubKeyAuthentication yes` signifie que par défaut le paramètre `PubKeyAuthentication` est à `Yes` et que donc l'authentification par clé est autorisée. Pour modifier un paramètre, il suffit donc de le décommenter et de définir sa valeur.

Les manipulations à effectuer sont :

1. modifier le numéro du port sur lequel le démon `ssh` est à l'écoute pour qu'il passe de `22` à `2024` ;
2. faire en sorte que le serveur utilise sa paire de clés ECDSA
3. désactiver l'authentification par mot de passe (naturellement cela n'a de sens que si les utilisateurs qui peuvent se connecter chez vous on effectivement mis en place une authentification par clé) ;
4. désactiver l'accès distant à votre compte `root` ;
5. après avoir rechargé la configuration du service `ssh` vous vérifierez que celle-ci correspond bien à ce qui est demandé.