TP5 – SSH

1. Introduction

Secure Shell (SSH) est à la fois un programme informatique et un protocole de communication sécurisé qui permet de se connecter à distance a une machine Linux.

Ainsi, dans ce TP, nous allons découvrir comment se connecter à distance à une machine sous Linux.

Prenons un cas concret : votre serveur au boulot est sous Linux. En fin de la journée chez vous, vous avez besoin de lancer un téléchargement ou de récupérer un document. Vous vous connectez à distance sur votre machine et vous ouvrez une console comme si vous étiez en face de votre machine serveur au boulot ! Tout ce que vous avez appris à faire dans une console, vous pouvez le faire à distance depuis n'importe quelle machine dans le monde.

Le protocole de connexion Secure Shell (SSH) impose un échange de clés de chiffrement en début de connexion. Par la suite, tous les segments TCP sont authentifiés et chiffrés. Il devient donc impossible d'utiliser un sniffer pour voir ce que fait l'utilisateur.

Le protocole SSH a été conçu avec l'objectif de remplacer les différents protocoles non chiffrés comme telnet, ftp, rcp, et rsh.

2. Prérequis

- ✓ Avoir terminer le TP4
- ✓ Une machine serveur sous Ubuntu Server 16.04 sur laquelle le serveur SSH sera installé, de préférence choisir l'une des machines serveur déjà configurée.
- ✓ Une machine cliente sous Windows ou Linux Desktop pour le teste du serveur SSH

3. Installation du serveur SSH

Pour installer le serveur SSH , on va installer la suite OpenSSH.

OpenSSH est un ensemble d'outils informatiques libres permettant des communications sécurisées sur un réseau informatique en utilisant le protocole SSH.

Taper la commande suivante pour installer le serveur SSH

```
root@ubuntu:~# apt-get update
```

```
root@ubuntu:~# apt-get install openssh-server
```

Lors de l'installation, vous devriez voir certaines étapes intéressantes s'effectuer automatiquement :

```
Creating SSH2 RSA key; this may take some time ...
2048 SHA256:8AO1K9jjIf1RCelIf3NdSBjoiix2FPZLVKDwUpYi3Is root@ubuntu (RSA)
Creating SSH2 DSA key; this may take some time ...
1024 SHA256:ZTJy4sNmPpTKDdoE0tJhFC2ntYZ9ezUPiDk+qXn+fzw root@ubuntu (DSA)
Creating SSH2 ECDSA key; this may take some time ...
256 SHA256:oBppuBu7i+LR4MDrvtQDd4u/OTWV7Wm2RcL6KCIv/xg root@ubuntu (ECDSA)
Creating SSH2 ED25519 key; this may take some time ...
256 SHA256:/G6aZiblO4sNN2sEOomMaWliuI68yT9LrYOdaZN3MRw root@ubuntu (ED25519)
```

RSA, DSA, ECDSA et ED25519 sont des algorithmes de chiffrement asymétrique. Comme je vous l'ai dit plus tôt, SSH peut travailler avec plusieurs algorithmes de chiffrement différents.

Ce que vous voyez là est l'étape de création d'une clé publique et d'une clé privée pour chacun des algorithmes (RSA, DSA, ECDSA et ED25519).

Normalement, le serveur SSH sera lancé à chaque démarrage. Si ce n'est pas le cas, vous pouvez le lancer à tout moment avec la commande suivante :

```
root@ubuntu:~# /etc/init.d/ssh start
[ ok ] Starting ssh (via systemctl): ssh.service.
```

4. Sécurisation du serveur SSH :

a) Modifier le port d'écoute :

La première chose à faire lors de l'installation d'un serveur SSH, est de changer le port d'écoute du serveur.

Localisez la ligne suivant dans le fichier **/etc/ssh/sshd_config** (généralement située au début de celui-ci) :

```
# Package generated configuration file
# See the sshd_config(5) manpage for details

# What ports, IPs and protocols we listen for
Port 220
```

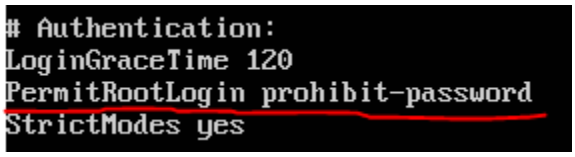
Remplacer la valeur du port 22 par 220 par exemple.

Ce changement permet de se prémunir des attaques de robots qui vont essayé de forcer votre serveur SSH sur le port par défaut qui est le 22.

b) Interdire les connexions à distance en root :

Une sécurisation supplémentaire est d'interdire les connexions directes pour l'utilisateur root en utilisant un mot de passe.

Cela se passe dans le même fichier que précédemment à l'aide de la ligne suivante :



Il faut s'assurer que PermitRootLogin est sur la valeur prohibit-password. Ainsi pour se connecter a distance avec l'utilisateur Root, on doit avoir une clé de déchiffrement (On voir voir comment se connecter a l'aide d'une clé plus tard dans le TP)

Sinon, vous pouvez interdire carrément la connexion avec le Root.

Pour se faire, on remplace **prohibit-password** par **no**. Cela ajoute une sécurité supplémentaire car même si l'accès à votre serveur SSH est forcé, il le sera à l'aide d'un utilisateur normal.

Pour l'instant, on va laisser **PermitRootLogin** en mode **prohibit-password**

Enregistrez le fichier modifié et redémarrez le serveur ssh avec la commande :

c) Ajouter un compte utilisateur

Utiliser la commande suivante pour ajouter un nouveau compte d'utilisateur :

sudo adduser nom_utilisateur

5. Se connecter au serveur SSH :

a) Se connecter a partir d'une machine cliente Linux :

Note ***: Afin d'avoir une connexion entre les deux machines(serveur SSH et cliente), assurez-vous que la machine serveur SSH est la machine cliente sont dans le même VMnet.***

Assurez-vous aussi que les deux machines ont obtenu une adresse IP du serveur DHCP. Sinon, assigner des adresses IP au deux machines manuellement.

Si vous vous connectez depuis un réseau externe, il vous faut entrer l'IP internet de la machine serveur que vous pouvez obtenir en allant sur www.whatismyip.com par exemple.

Toutes les machines équipées de Linux proposent la commande ssh qui permet de se connecter à distance à une autre machine.

Sur la machine cliente, utilisez la commande ssh comme ceci :

ssh nom_utilisateur@votre_serveur -p votre_numéro_de_port

Assurer vous que le compte utilisateur existe déjà dans la machine cliente, sinon créer un nouveau compte d'utilisateur.

```
root@ubuntu:~# ssh mohamed@192.168.1.24 -p 220
```

Normalement, le serveur devrait répondre au bout d'un moment et vous devriez voir quelque chose comme ce qui suit :

```
The authenticity of host '[192.168.1.24]:220 ([192.168.1.24]:220)' can't be established.
ECDSA key fingerprint is SHA256:oBppuBu7i+LR4MDrvtQDd4u/OTWV7Wm2RcL6KCIv/xg.
Are you sure you want to continue connecting (yes/no)? y
Please type 'yes' or 'no': yes
```

Que se passe-t-il ? On vous dit que le fingerprint (empreinte) du serveur est `oBppuBu7i+LR4MDrvtQDd4u/OTWV7Wm2RcL6KCIv/xg.`. C'est un numéro unique qui vous permet d'identifier le serveur. Si demain quelqu'un essaie de se faire passer pour le serveur, le fingerprint changera forcément et vous saurez qu'il se passe alors quelque chose d'anormal. Ne vous inquiétez pas, SSH vous avertira de manière très claire si cela arrive.

En attendant, tapez « yes » pour confirmer que c'est bien le serveur auquel vous voulez vous connecter. Le serveur et le client vont alors s'échanger une clé de chiffrement, comme je vous l'ai expliqué un peu plus tôt. Normalement, le serveur devrait vous demander au bout de quelques secondes votre mot de passe.

Dans l'exemple si dessus, je me suis connecté en tant que l'utilisateur **mohamed** sur la machine serveur **192.168.1.24** sur le port **220.**

Si vous entrez le bon mot de passe, la console du serveur devrait vous afficher un message de bienvenue puis un prompt qui correspond à la console du serveur. Bravo, vous êtes connectés !

```
mohamed@192.168.1.24's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-87-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
Last login: Sat Mar 31 06:44:33 2018
mohamed@ubuntu:~$
```

Si la connexion ne passe pas, vérifiez vos paramètres.

Pour vous déconnecter, tapez **logout** ou son équivalent : la combinaison de touches Ctrl + D.

```
mohamed@ubuntu:~$ logout
Connection to 192.168.1.24 closed.
root@ubuntu:~# _
```

b) Se connecter à partir d'une machine cliente Windows :

Note : Dans cette étape, je vais utiliser une machine virtuelle windows 7 configurée sur le même VMnet que la machine serveur SSH.

Vous pouvez aussi utiliser la machine hôte en tant que machine cliente.

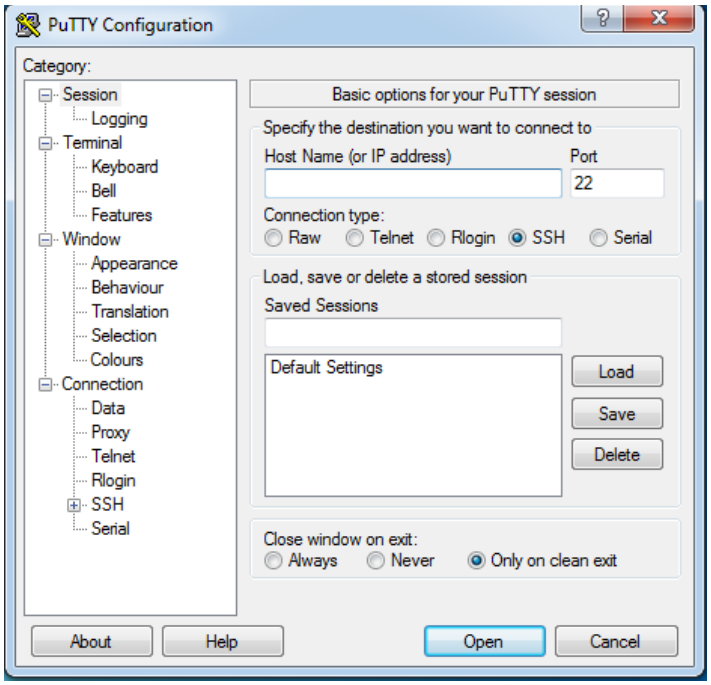
Assurez-vous aussi que les deux machines ont obtenu une adresse IP du serveur DHCP. Sinon, assigner des adresses IP au deux machines manuellement.

Il existe des programmes pour Windows si voulez avoir accès à la console de votre machine serveur Linux. Le plus connu d'entre eux – celui que j'utilise personnellement – s'appelle PuTTY.

Vous pouvez télécharger PuTTY depuis son site officiel.

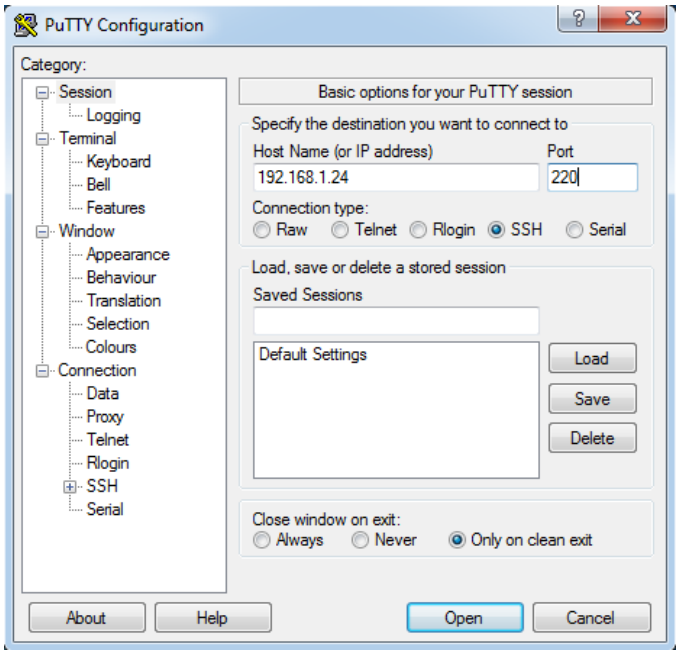
https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

Une fois que c'est fait et installé, lancez PuTTY. Une fenêtre comme celle de la figure suivante devrait s'afficher.

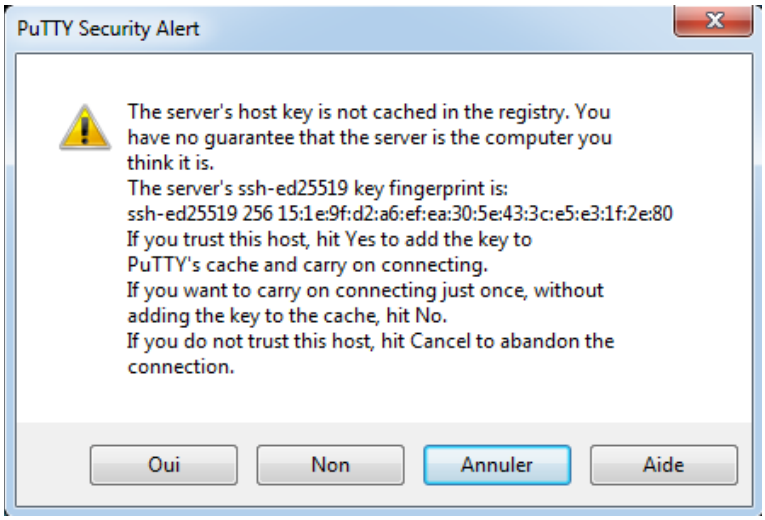


Il y a beaucoup de pages d'options, comme vous pouvez le voir au niveau de la section « Category » sur le côté. Pour le moment, vous avez juste besoin de remplir le champ en haut « Host Name (or IP address) » ainsi que le numeoro de port, vu qu'on a changé le port par défaut. Entrez-y l'adresse IP de votre ordinateur sous Linux aisni que le port.

Dans ce cas, je vais entrer l'adresse IP de mon PC Linux (serveur ssh)situé sur le meme VMnet que la machine windows (192.168.1.24) et le port (220)



La première fois que vous vous connectez à votre serveur, PuTTY devrait vous demander une confirmation comme sur la figure suivante.



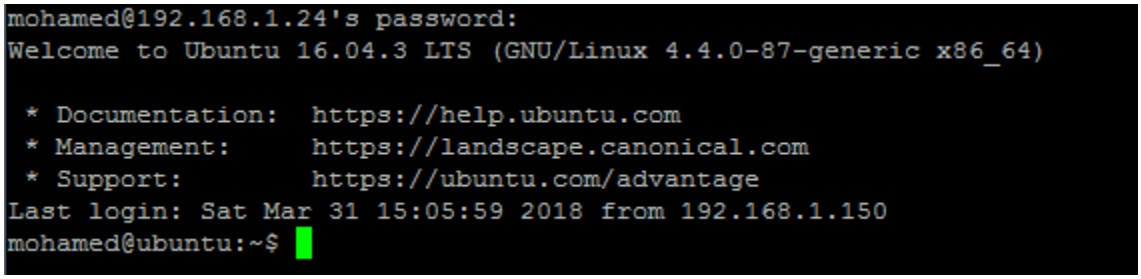
C'est la même chose que sous Linux : on vous donne l'empreinte (fingerprint) de votre serveur. Vous devez confirmer que c'est bien chez lui que vous voulez vous connecter. Cliquez sur « Oui » pour confirmer.

À l'avenir, on ne vous reposera plus la question. Par contre, si le fingerprint change, un gros message d'avertissement s'affichera. Cela signifiera soit que le serveur a été réinstallé, soit que quelqu'un est en train de se faire passer pour le serveur.

La le serveur vous demande, « vous voulez se connecter en tant que » et saisissez le nom du compte auquel vous voulez se connecter.

Dans ce cas j'ai choisi l'utilisateur **mohamed.**

Entrez ensuite le mot de passe la console du serveur devrait vous afficher un message de bienvenue puis un prompt qui correspond à la console du serveur. Bravo, vous êtes connectés !



Pour vous déconnecter, tapez logout ou son équivalent : la combinaison de touches Ctrl + D.

6. L'identification automatique par clé :

Il y a plusieurs façons de s'authentifier sur le serveur, pour qu'il sache que c'est bien vous. Les deux plus utilisées sont :

- l'authentification par mot de passe ;
- l'authentification par clés publique et privée du client.

Pour le moment, nous avons vu uniquement l'authentification par mot de passe (le serveur vous demandait votre mot de passe).

Il est possible d'éviter que l'on vous le demande à chaque fois grâce à une authentification spéciale par clé. Cette méthode d'authentification est plus complexe à mettre en place, mais elle est ensuite plus pratique.

Avec cette nouvelle méthode d'authentification, c'est le client qui va générer une clé publique et une clé privée. Les rôles sont un peu inversés.

L'avantage, c'est que l'on ne vous demandera pas votre mot de passe à chaque fois pour vous connecter. Si vous vous connectez très régulièrement à un serveur, c'est vraiment utile. Si vous faites bien les choses, cette méthode est tout aussi sûre que l'authentification par mot de passe.

a) Authentification par clé depuis Linux

Pour mettre en marche ce mode d'authentification, nous allons d'abord devoir effectuer des opérations sur la machine du client, puis nous enverrons le résultat au serveur.

• Opérations sur la machine cliente

Il faut tout d'abord vous rendre sur la machine cliente et taper la commande suivante pour générer une paire de clés publique / privée :

```
root@ubuntu:~# ssh-keygen -t rsa
```

Vous pouvez remplacer rsa par dsa si vous voulez utiliser l'autre algorithme de chiffrement, mais ça n'a pas vraiment d'importance ici.

Lorsque vous tapez cette commande, vous allez voir plusieurs messages s'afficher et il vous sera demandé quelques petites précisions :

Dans un premier temps, le client génère une paire de clés (« Generating public/private rsa key pair »).

Il doit ensuite sauvegarder ces clés dans des fichiers (un pour la clé publique, un pour la clé privée). On vous propose une valeur par défaut : je vous conseille de ne rien changer et de taper simplement Entrée.

```
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
```

Ensuite, on vous demande une passphrase. C'est une phrase de passe qui va servir à chiffrer la clé privée pour une meilleure sécurité. Là, vous avez deux choix :

- soit vous tapez Entrée directement sans rien écrire, et la clé ne sera pas chiffrée sur votre machine ;
- soit vous tapez un mot de passe de votre choix, et la clé sera chiffrée.

Tout le monde ne met pas une phrase de passe. En fait, ça dépend du risque que quelqu'un d'autre utilise la machine du client et puisse lire le fichier contenant la très secrète clé privée. Si le PC du client est votre PC chez vous et que personne d'autre ne l'utilise, il y a assez peu de risques (à moins d'avoir un virus, un spyware…). Si c'est en revanche un PC public, je vous recommande vivement de mettre une passphrase pour chiffrer la clé qui sera enregistrée.

Si vous hésitez entre les deux méthodes, je vous recommande de rentrer une passphrase : c'est quand même la méthode la plus sûre.

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:Q3TE0ltziwABUCkDeVV0GSCWJQdbZKTzBvh+S03aVvs root@ubuntu
The key's randomart image is:
+---[RSA 2048]----+
|  .o.+O@#B=o     |
|  . +ooX.o= o .  |
|   ..o= .. + + . |
|     . =  . . .  |
|      . S . .    |
|     . . * . .   |
|      . + + .    |
|       o o   .   |
|         .    E  |
+----[SHA256]-----+
root@ubuntu:~# _
```

Il faut maintenant envoyer au serveur votre clé publique pour qu'il puisse vous chiffrer des messages.

Votre clé publique devrait se trouver dans ~/.ssh/id_rsa.pub .
Votre clé privée, elle, se trouve dans ~/.ssh/id_rsa.

Vous pouvez lister le contenu du dossier .ssh, pour commencer :

Si vous faites un ls, vous devriez voir ceci :

```
root@ubuntu:~# ls ~/.ssh/
id_rsa  id_rsa.pub  known_hosts
```

Les trois fichiers sont :

- **id rsa :** votre clé privée, qui doit rester secrète. Elle est chiffrée si vous avez rentré une passphrase ;
- **id rsa.pub :** la clé publique que vous pouvez communiquer à qui vous voulez, et que vous devez envoyer au serveur ;
- **known hosts :** c'est la liste des fingerprint que votre PC de client tient à jour. Ça lui permet de se souvenir de l'identité des serveurs et de vous avertir si, un jour, votre serveur est remplacé par un autre (qui pourrait être celui d'un pirate !).

Maintenant on va envoyer la clé publique au serveur ssh. Nous travaillons toujours sur la machine cliente.

L'opération consiste à envoyer la clé publique (id_rsa.pub) au serveur et à l'ajouter à son fichier authorized_keys (clés autorisées). Le serveur y garde une liste des clés qu'il autorise à se connecter.

Le plus simple pour cela est d'utiliser la commande spéciale **ssh-copy-id**. Utilisez-la comme ceci :

```
root@ubuntu:~# ssh-copy-id -i ~/.ssh/id_rsa.pub  mohamed@192.168.1.24 -p 220
```

Vous serez ensuite inviter a entre le mot de passe du compte auquel vous voulez se connecter (dans cet exemple celui du compte mohamed sur la machine serveur ssh), si tout se passe bien, vous devez voir le message suivant indiquant l'ajout d'une clé :

```
mohamed@192.168.1.24's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh -p '220' 'mohamed@192.168.1.24'"
and check to make sure that only the key(s) you wanted were added.
```

Maintenant, connectez-vous au serveur comme vous le faisiez auparavant :

```
root@ubuntu:~# ssh mohamed@192.168.1.24 -p 220
```

Si vous avez choisi une passphrase lors du chiffrement de la clé, vous serez invité à l'entrer, sinon, vous allez vous connecter automatiquement.

Afin d'éviter d'entrer la passphrae a chaque fois, on va utiliser l'agent SSH qui est un programme qui tourne en arrière-plan en mémoire. Il retient les clés privées pendant toute la durée de votre session.

Tout ce que vous avez à faire est de lancer le programme ssh-add sur le PC du client avec la commande suivante:

ssh-add

Il va automatiquement chercher votre clé privée. Pour la déchiffrer, il vous demande la passphrase. Entrez-la.

Maintenant que c'est fait, chaque fois que vous vous connecterez à un serveur, vous n'aurez plus besoin d'entrer la passphrase. Essayez de vous connecter à votre serveur pour voir !

b) Authentification par clé depuis Windows (PuTTY)

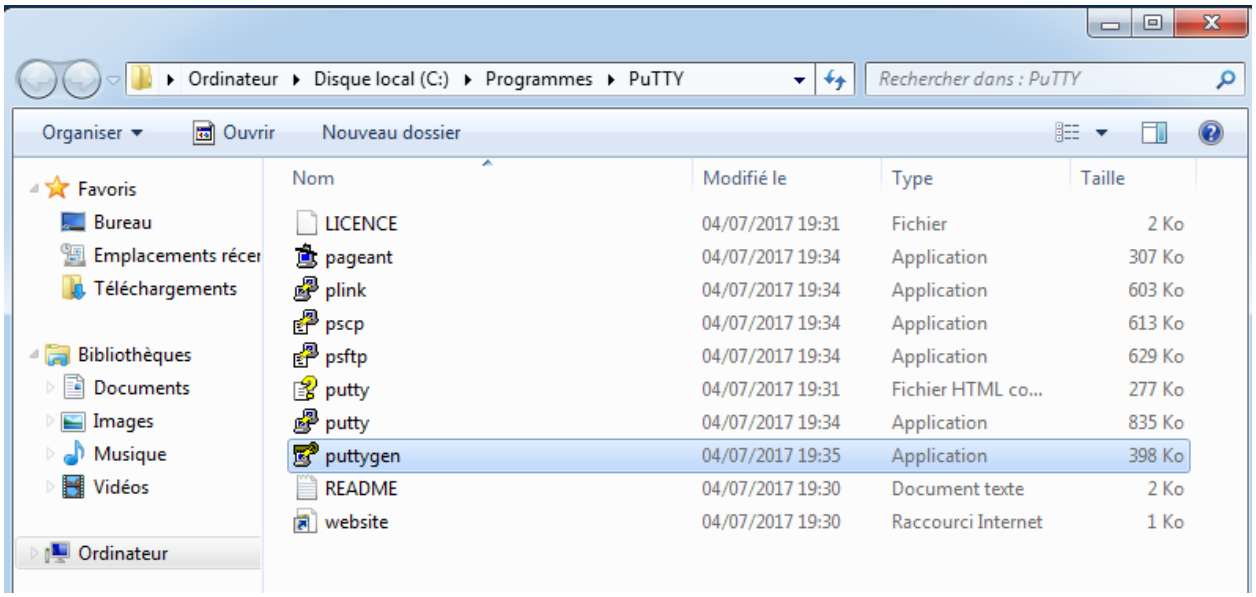
Il est tout à fait possible d'utiliser l'authentification par clé avec PuTTY.

Le principe est le même que sous Linux : il faut d'abord que l'on génère une paire de clés sur le PC du client, puis qu'on les envoie au serveur. Nous retrouverons aussi un équivalent de l'agent SSH pour éviter d'avoir à entrer une passphrase à chaque fois.

Commençons par la génération des clés.

- **Générer une paire de clés (publique et privée) avec Puttygen**

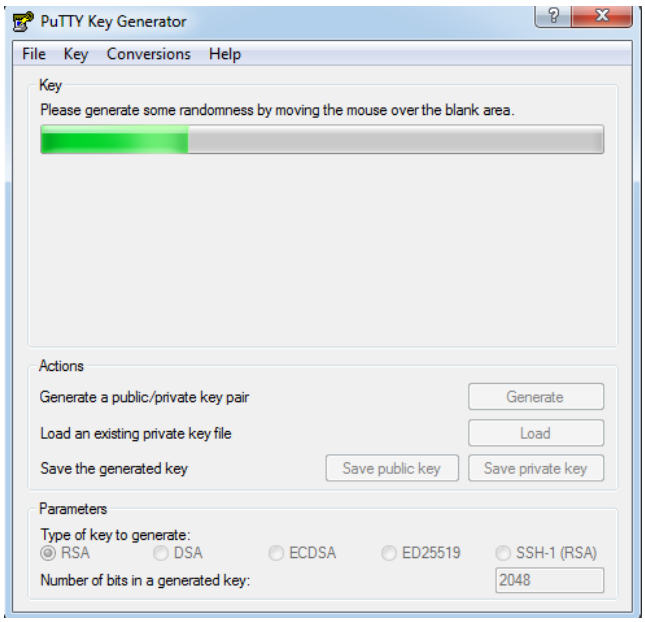
Normalement, vous devriez avoir installé un programme appelé Puttygen (figure suivante – il se trouvait dans le gestionnaire d'installation de PuTTY). Lancez-le.



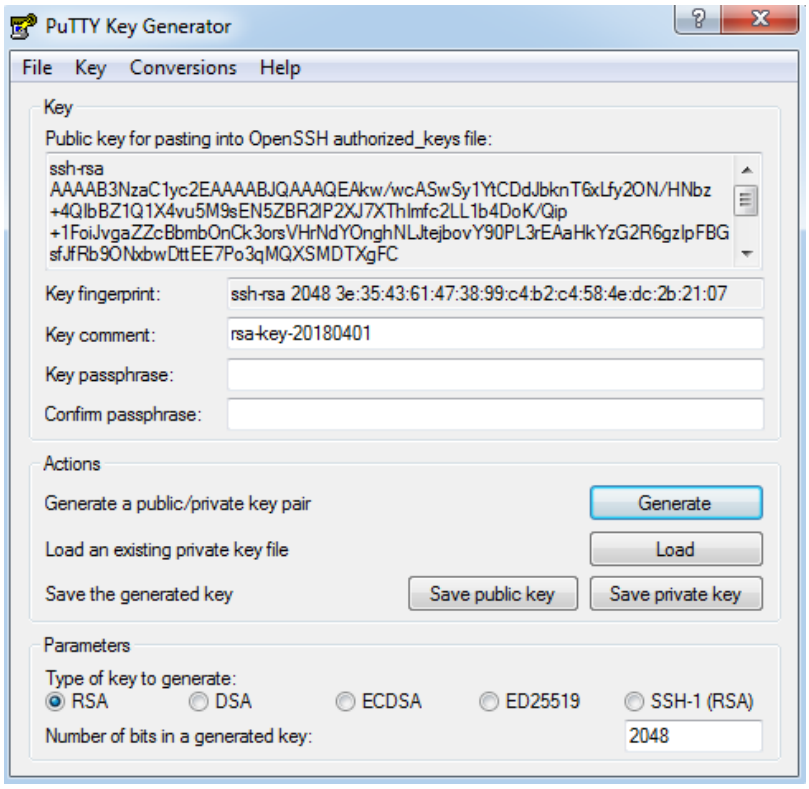
En bas de la fenêtre, vous pouvez choisir vos paramètres : algorithme de chiffrement et puissance du chiffrement. Les valeurs par défaut (ici RSA 2048 bits) sont tout à fait convenables. Vous pouvez les changer, mais sachez qu'elles sont sûres et que vous pouvez vous en contenter.

Cliquez sur le bouton « Generate ». Le programme va générer une paire de clés (publique et privée).

Pour l'aider à générer cette paire, le programme vous propose quelque chose d'assez amusant : vous devez bouger la souris dans la fenêtre (figure suivante)



Une fois que c'est fait, on vous affiche la clé publique (figure suivante) :



Comme vous le voyez, cela ne me dérange pas que tout le monde voie ma clé publique. Le principe, c'est justement que tout le monde peut voir cette clé, mais ne peut rien en faire. Par contre, la clé privée doit rester secrète.

Vous pouvez choisir d'entrer une passphrase ou non. Comme je vous l'ai expliqué plus tôt, cela renforce la sécurité en chiffrant la clé privée.

Saisissez la passphrase dans les champs « Key passphrase » et « Confirm passphrase ».

Ensuite, enregistrez la clé publique dans un fichier en cliquant sur « Save public key ». Donnez-lui l'extension **.pub**. Vous pouvez nommer ce fichier comme vous voulez, par exemple **publicKey.pub**. Enregistrez-le où vous voulez.

Puis enregistrez la clé privée en cliquant sur « Save private key ». Donnez-lui l'extension **.ppk** : **privateKey.ppk** par exemple.

Ne fermez pas encore Puttygen.

- **Envoyer la clé publique au serveur**

Comme sous Linux tout à l'heure, il faut envoyer la clé publique au serveur pour qu'il nous autorise à nous connecter par clé. Le problème, c'est qu'il n'y a pas de commande pour le faire automatiquement depuis Windows. Il va falloir ajouter la clé à la main dans le fichier authorized_keys. Heureusement, ce n'est pas très compliqué.

Ouvrez PuTTY et connectez-vous au serveur comme auparavant (en entrant votre mot de passe habituel). Rendez-vous dans ~/.ssh :

```
mohamed@ubuntu:~$ cd ~/.ssh
mohamed@ubuntu:~/.ssh$
```

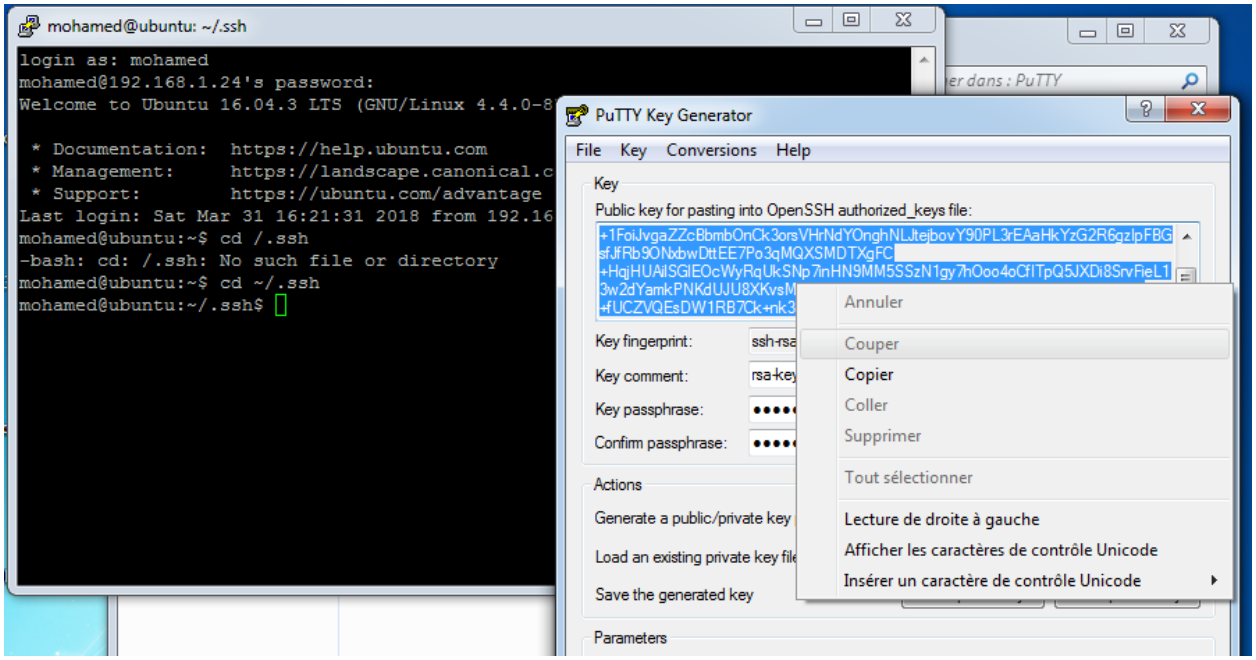
Si le dossier .ssh n'existe pas, pas de panique, créez-le :

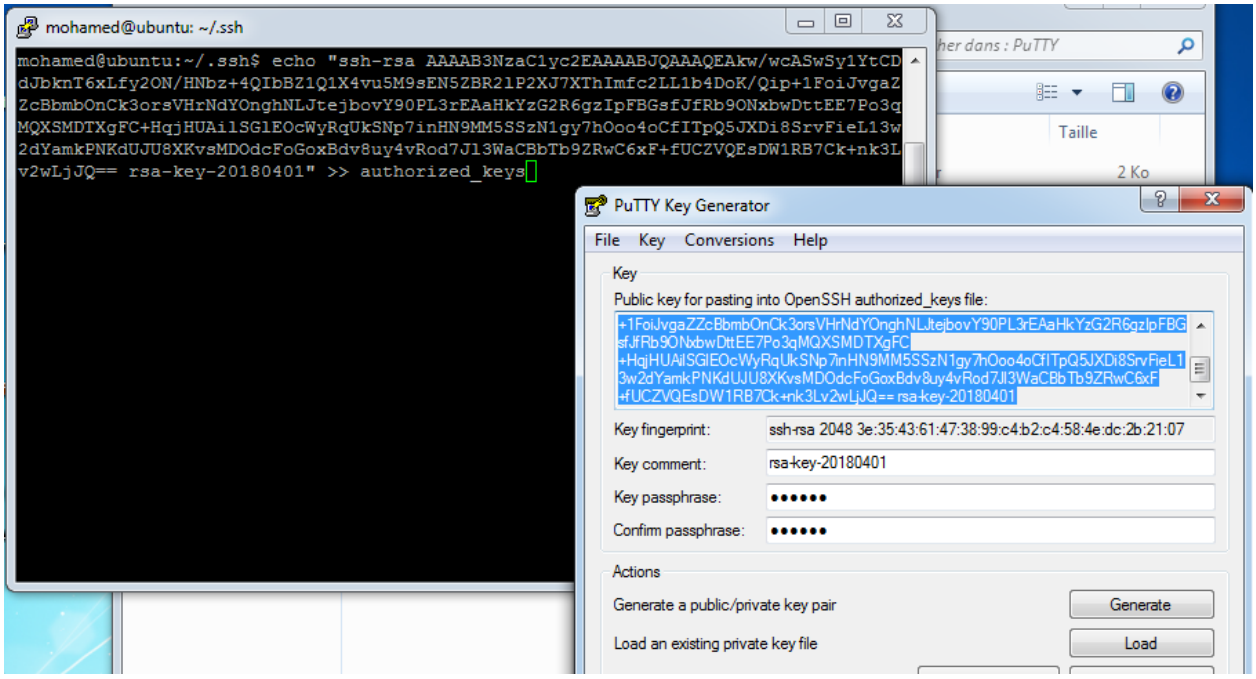
mkdir .ssh

Rajoutez votre clé publique à la fin du fichier authorized_keys (s'il n'existe pas, il sera créé). Vous pouvez utiliser la commande suivante :

echo "votre_cle_publique" >> authorized_keys

votre clé publique est affichée dans Puttygen, que vous ne devriez pas avoir fermé. Pour coller la clé dans la console, utilisez la combinaison de touches Shift + Inser plutôt que Ctrl + V.



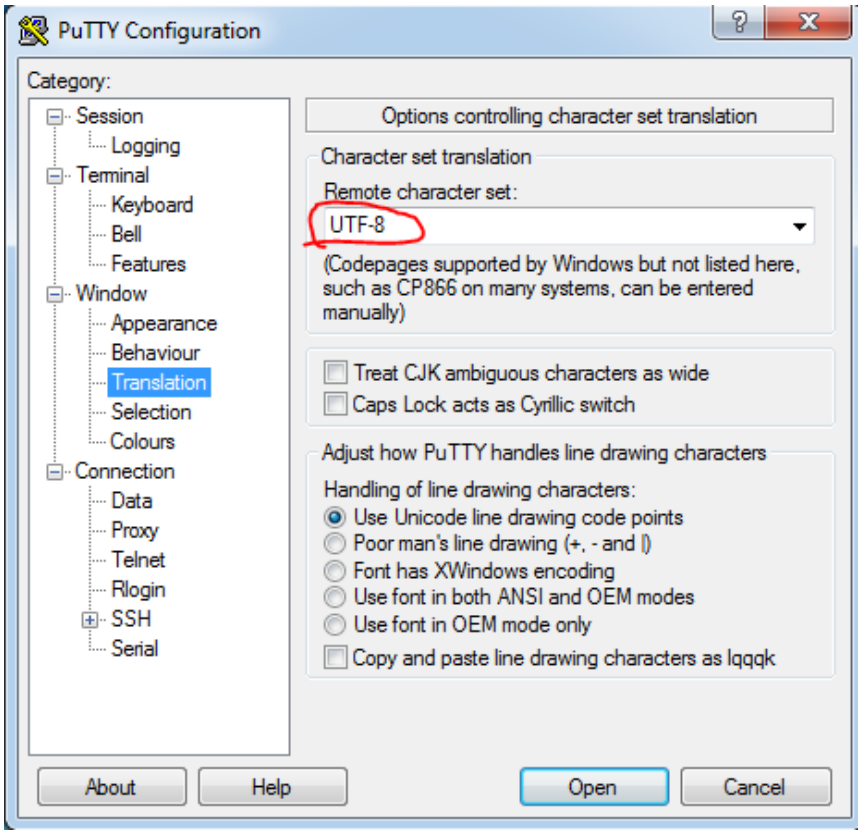


Déconnectez-vous, et relancez PuTTY. On va maintenant le configurer pour qu'il se connecte à l'aide de la clé.

- **Configurer PuTTY pour qu'il se connecte avec la clé**

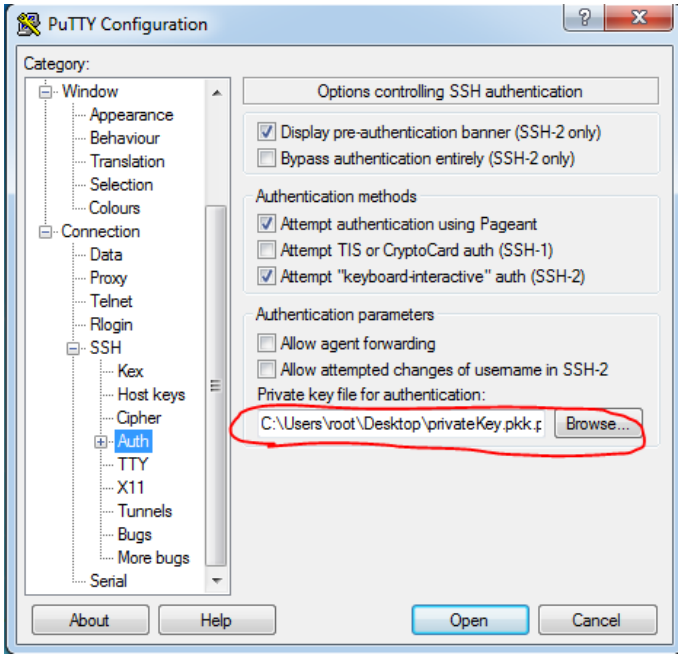
Une fois PuTTY ouvert, rendez-vous dans la section Window → Translation pour commencer. Ça n'a pas de rapport direct avec les clés, mais cela vous permettra de régler le problème des accents qui s'affichent mal dans la console si vous l'avez rencontré.

Réglez la valeur de la liste déroulante à UTF-8, comme sur figure suivante.

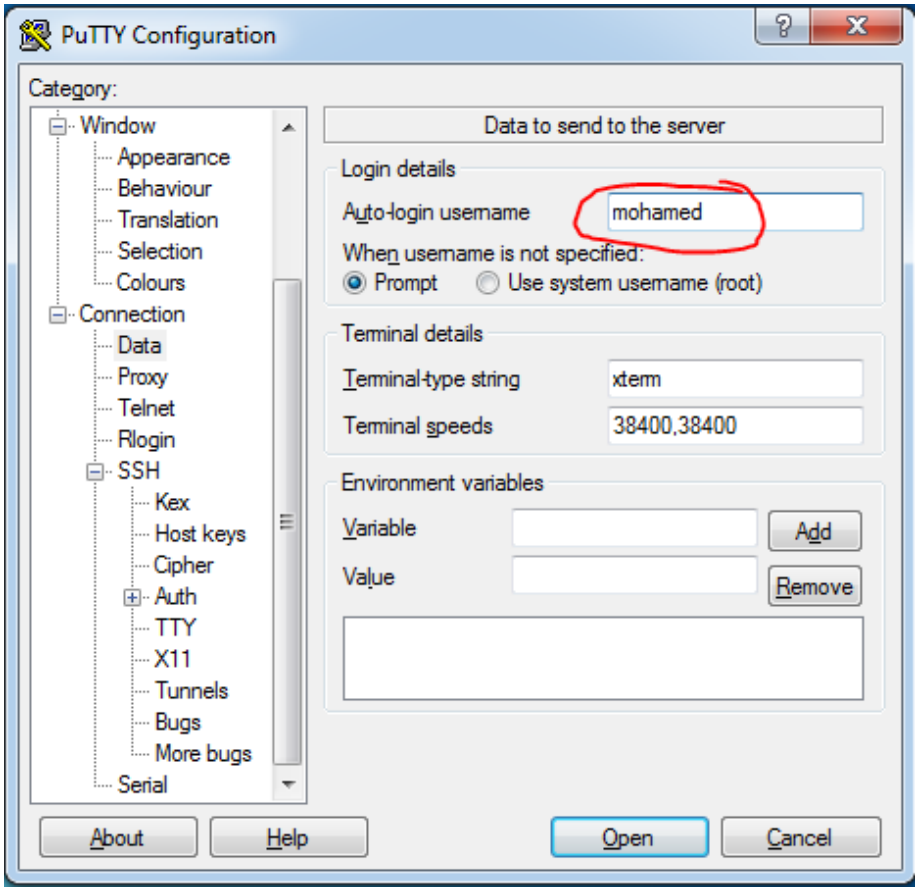


La plupart des serveurs encodent désormais les caractères en UTF-8, cela devrait donc vous éviter des soucis d'affichage.

Maintenant, rendez-vous dans Connection → SSH → Auth. Cliquez sur le petit bouton « Browse » pour sélectionner votre clé privée (figure suivante).

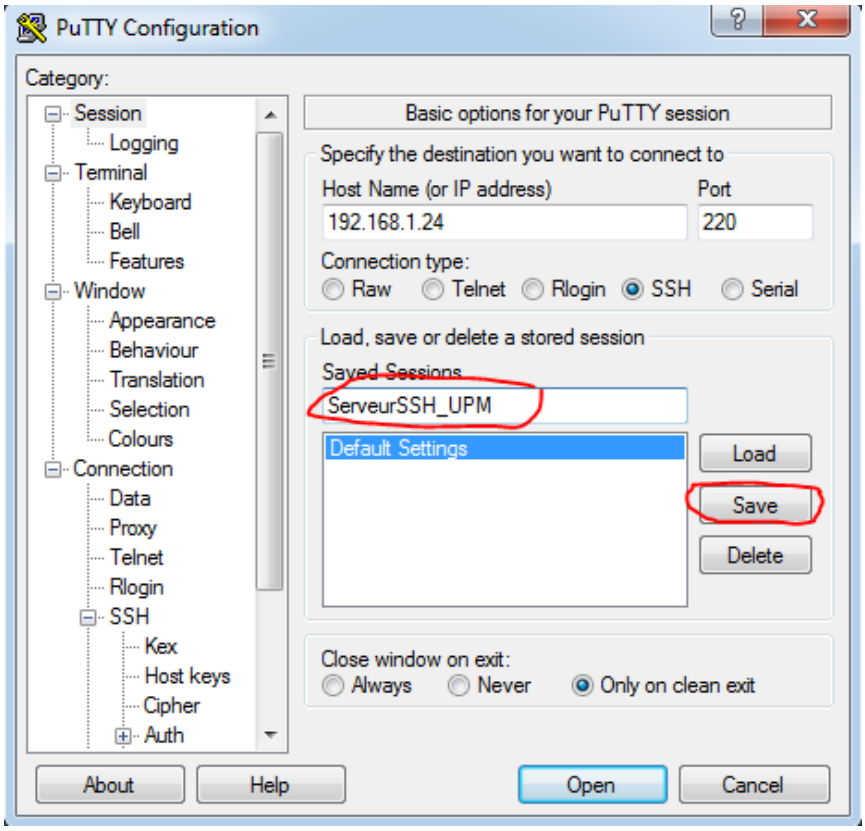


Je vous recommande aussi d'aller dans Connection → Data et d'entrer votre login dans « Auto-login username », comme la figure suivante vous le montre.



Retournez à l'accueil en cliquant sur la section « Session » tout en haut (figure suivante). Entrez l'IP du serveur.

Ensuite, je vous recommande fortement d'enregistrer ces paramètres.



Donnez un nom à votre serveur (par exemple ServeurSSH_UPM) sous « Saved Sessions ». Cliquez ensuite sur « Save ».

À l'avenir, vous n'aurez qu'à double-cliquer sur le nom de votre serveur dans la liste pour vous y connecter directement avec les bons paramètres.

Cliquez sur « Open » pour vous connecter au serveur.

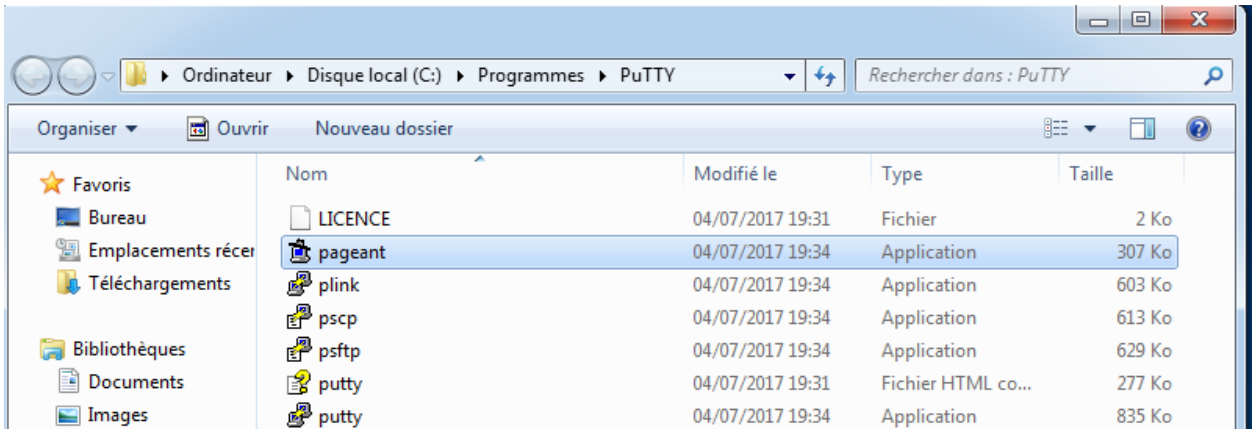
Vous devriez voir PuTTY utiliser automatiquement votre pseudo.

Si vous avez choisi une passphrase, il va vous demander votre passphrase. Entrez-la pour vérifier que cela fonctionne.

Et si je ne veux pas avoir à entrer la passphrase à chaque fois ?

- **L'agent SSH Pageant**

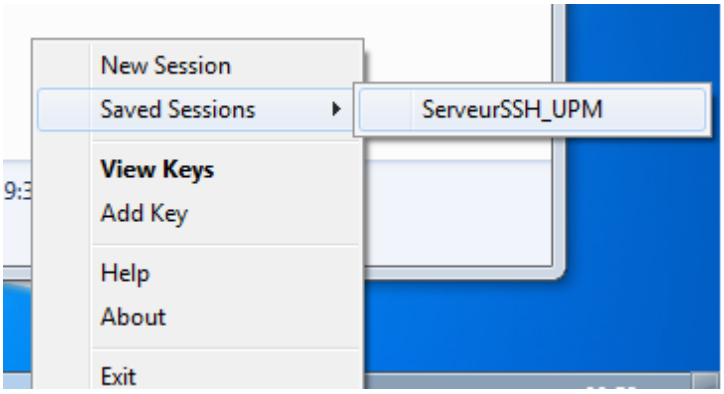
L'agent SSH installé avec PuTTY s'appelle « Pageant ». Je vous recommande de le lancer au démarrage de l'ordinateur automatiquement



Lorsque vous lancez « Pageant », la petite icône d'un ordinateur avec un chapeau s'ajoute dans la barre des tâches à côté de l'horloge, comme sur la figure suivante.

Faites un clic droit dessus, puis cliquez sur « Add key ». On vous demande où se trouve la clé privée (privateKey.ppk). Entrez ensuite la passphrase.

C'est bon. Vous avez juste besoin de le faire une fois. Maintenant, vous pouvez vous connecter au serveur que vous voulez en faisant un clic droit sur l'icône, puis en sélectionnant « Saved Sessions » (figure suivante).



Notez que si l'agent SSH « Pageant » est pratique, il vaut mieux l'arrêter si vous devez vous absenter de votre ordinateur un long moment et que quelqu'un risque de l'utiliser. Sinon, n'importe qui peut se connecter à vos serveurs sans avoir à entrer de mot de passe.

Retenez bien : l'agent SSH est un compromis entre la sécurité et le côté pratique. Il retient les clés pour vous (du moins tant que le programme tourne). Si vous êtes des utilisateurs intensifs de SSH, cela vous fera gagner beaucoup de temps.