Télétravail avec `ssh`

Objectifs

Simplifier considérablement tous les accès aux serveurs & services à distance quand ce n'est pas tout simplement autoriser l'accès aux services (parfois il n'y a pas d'autre solution). Ce peut être :

- simple connexion *ssh*, travail direct sur les serveurs du laboratoire, de calcul (« calculco »), de la PLM etc.
- transferts de fichiers : *scp*
- sauvegardes : *rsync*, *unison* etc.
- connexions plus complexes : *rebonds*
- tunnel ssh, proxy : accès aux serveurs de jeton (matlab, maple), accès au serveur « charges des services »
- autres services : git, … ?

Les exemples de cette fiche supposent que l'utilisatrice Agathe Zeblouse a pour identifiants `agathe` et `zeblouse` respectivement sur son portable et au travail.

1 Principes et généralités

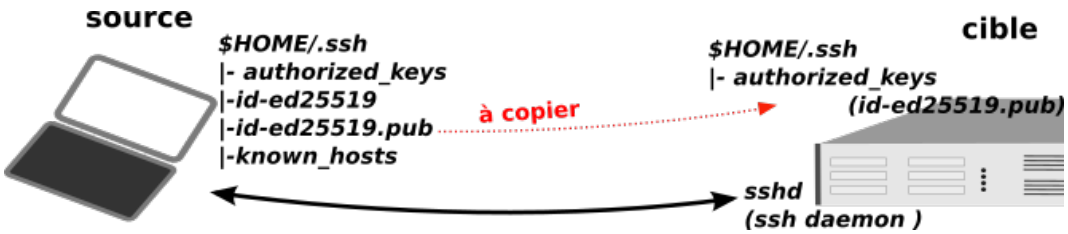

FIGURE 1 – ssh rep et fichiers

Remarque : sur un UNIX (Mac OS, Linux) tous les les fichiers se trouvent dans le répertoire caché (car le nom du répertoire commence par un « . ») `.ssh` de votre répertoire d'accueil. Pour vérifier que les droits sont corrects :

```
agathe@portable:~\$ ls -a (-a pour afficher les fichiers/rep. cachés)
...
drwx------  2 agathe test  4096 mars   7 12:14 .ssh
```

Ici le répertoire n'est bien accessible qu'à l'utilisateur agathe, la clef privée Vous pouvez supprimer ce répertoire et son contenu sans conséquence (si ce n'est d'avoir à recommencer les manipulations suivantes :-))

1.1 Étape1 : générer le couple de clefs publique/privée

Dans un terminal, taper la commande :

```
ssh-keygen -t ed25519 -o -a 64  # si votre client ne supporte
                                # pas ed25519 cf. note ci-dessous

Generating public/private ed25519 key pair.
file in which to save the key (/home/agathe/.ssh/id_ed25519)
passphrase (empty for no passphrase)
same passphrase again:

Your identification has been saved in /home/agathe/.ssh/id_ed25519.
Your public key has been saved in /home/agathe/.ssh/id_ed25519.pub.
The key fingerprint is:
+--[ED25519 256]--+
|          ..+.o. |
|.oo o    o o +.=.|
|o.o@ .. . o o = o|
|oo o o o . o .   |
|++++o.= S .      |
|...+.o + o       |
|E o   . . .      |
|..     o         |
|      .          |
+-----[SHA256]-----+
```

Commentaires sur la séquence

1. `id_ed25519` est le nom de la clé par défaut. Si vous utilisez intensivement ces méthodes de connexions (plusieurs identités, plusieurs institutions) il peut être judicieux de générer plusieurs couples de clefs. - passphrase : cf. **l'avertissement** ci-après.
2. `id_ed25519.pub` : nom de fichier de la clef publique qu'il faudra transmettre sur le(s) serveur(s) cible(s). - **RSA et elliptic curves** : si votre client ssh, trop vieux, ne supporte

pas la commande `ssh-keygen -t ed25519`, générer une clef *RSA* avec la commande `ssh-keygen -t rsa -b 4096`. En effet, pour une sécurité « correcte » avec *RSA* , il faut forcer une longueur supérieure à celle par défaut de votre système (qui est probablement 2048).

Attention

Il est important de bien comprendre que les manipulations précédentes entraînent des risques du point de vue sécurité : si une personne vous vole votre clef privée et que vous n'avez pas mis de passphrase, il a libre accès à votre compte sans mot de passe, depuis n'importe quelle machine. (il lui suffit -par exemple- de profiter d'une absence & d'une session ouverte pour copier les deux clefs privées/publiques.

- si cette clef vous sert sur plusieurs services/serveurs, cette personne est en possession dun passe-partout(!)
- ce système d'identification est complètement indépendant du login/passwd ... changer de mot de passe ne résoud pas le problème(!)
- néanmoins, avec une passphrase correcte, ces méthodes de connexions sans mot de passe, sont au contraire recommandées par certains : êtes vous sûr de la fiabilité du flux client-serveur lorsque vous tapez **n** fois votre mot de passe dans la journée ?

1.2 Étape 2 : copier la clef publique id_ed25519.pub sur l'hôte cible

Il suffit de taper la commande :

```
agathe@portable : ssh—copy—id —i ~/.ssh/id_ed25519.pub zeblouse@cible.fr
```

Que fait la commande précédente ? Elle ajoute simplement la clé publique dans le fichier `~/.ssh/authorized_keys` sur le compte zeblouse du serveur `cible.fr`.
Si la commande `ssh-copy-id` n'est pas disponible sur votre système :
Plusieurs méthodes sont possibles (cf. comment transférer un fichier).

Une solution élégante :

```
agathe@portable : cat ~/.ssh/id_ed25519.pub |ssh zeblouse@cible.fr "cat — >> ~/.ssh/authorized_keys"
```

On peut aussi bien entendu utiliser un éditeur pour ajouter la clef à la fin du fichier `authorized_keys`. (plusieurs clefs sont possibles pour le même compte cible)

Vous pouvez réitérer la manipulation sur tous les serveurs que vous utilisez fréquemment et vous connecter désormais sans mot de passe.

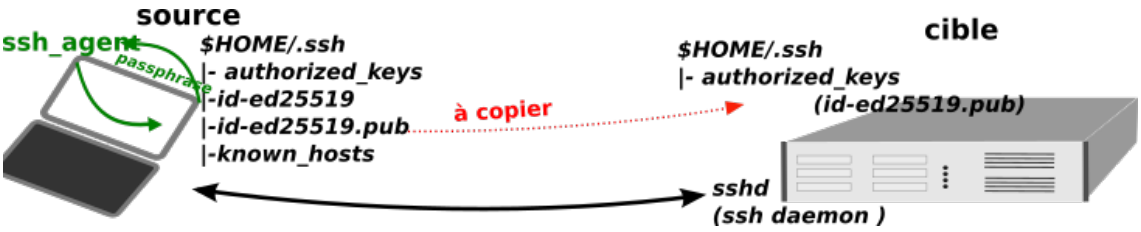

FIGURE 2 – agent ssh

1.3 Connexions sans mot de passe, avec passphrase

Cette version est la plus sérieuse du point de vue de sécurité.

Si vous avez une saisie une « passphrase » lorsque vous avez générez les deux clefs, alors vous êtes dans ce cas de figure. En effet, la méthode exige de renseigner la passphrase (une fois) à chaque session. Vous confiez la passphrase qui protège votre clef privée à « l'agent ssh » et ce dernier, autorisera alors toutes les connexions ssh vers les différentes cibles sans mot de passe.

Passer la main à l'agent ssh :

```
agathe@portable : ssh-add ~/.ssh/id_ed25519
Enter passphrase for /home/agathe/.ssh/id_ed25519 :
Identity added : /home/agathe/.ssh/id_ed25519
```

On peut vérifier que la clef privée est bien présente via la commande `ssh-add -l` ou supprimer des clefs par la commande `ssh-add -d <file>`.

Si l'agent ssh ne démarre pas automatiquement en même temps que votre PC, lancez-le avec la commande suivante :

```
agathe@portable : eval `ssh-agent`
```

À partir de ce moment, les commandes suivantes ne requièrent plus la saisie du mot de passe :

```
agathe@portable : scp fichier zeblouse@cible :
agathe@portable : ssh zeblouse@cible
zeblouse@cible :
```

Commentaires sur la séquence

- copie de `fichier` sur le répertoire d'accueil de `cible`
- se connecter sur `cible`

— … c'est fait! (pas de mot de passe saisi)

2 Astuces supplémentaires

2.1 les basiques (version non optimale)

L'efficacité est limitée, on peut néanmoins citer :

— choisir le même login sur vos portables vous affranchit de taper `zeblouse@`

— déclarer `univ-littoral.fr` comme domaine de recherche par défaut des propriétés DNS (propriétés réseaux) vous affranchit de taper `.univ-littoral.fr`

— fixer vous même le nom/alias des machines auxquelles vous vous connectez fréquemment dans le fichier `hosts` de votre portable vous affranchit également de taper `.univ-littoral.fr`. Pour les Unix-like (Linux, MacOs) le fichier est `/etc/hosts` et, pour Windows : `C:\Windows\System32\drivers\etc\hosts`.

2.2 configuration optimale : renseigner le fichier `~/.ssh/config`

au plus simple :

```
# contenu du \$HOME/.ssh/config de votre portable
Host piccolo
HostName piccolo.univ-littoral.fr
    User zeblouse
```

ou encore :

```
# contenu du \$HOME/.ssh/config de votre portable
Host pic
    HostName piccolo.univ-littoral.fr
    User zeblouse
    IdentityFile /home/agathe/.ssh/labo
```

Commentaires sur la séquence

— *Host* est un alias : on peut mettre n'importe quoi

— *User* : facultatif si login(source)= login(cible), ce qui n'est pas le cas dans nos exemples (agathe ≠ zeblouse)

— Agathe à l'étape un a généré *labo* et *labo.pub* (au lieu du couple par défaut id_ed25519 et id_ed25519.pub), on doit forcer l'usage de la bonne clef.

La connexion s'établie alors simplement avec la commande :

```
agathe@portable : ssh pic
```

Vous pouvez multiplier les serveurs dans ce fichier `config`

3 ssh : exemples pratiques

Lorsque vous n'êtes pas sur le « bon » réseau (extérieur, EDUROAM…), ssh vous permet d'accéder malgré tout à des services (par l'établissement d'un *tunnel ssh*) ou des machines (par *rebonds*) qui sont soit masqués par le pare-feu de l'université, soit simplement disponibles que sur le réseau local du laboratoire.

Raccourci

Si vous êtes déjà un utilisateur confirmé de ces fonctionnalités, passez rapidement les exemples et consultez directement la dernière section de cet article qui vous indiquera comment configurer votre fichier `~/.ssh/config` afin de bénéficier de l'ensemble des fonctionnalités des exemples.

3.1 Exemple 1 (tunnel) : imprimer depuis le réseau eduroam

Vous êtes au labo., connecté sur le réseau EDUROAM : vous n'avez pas accès au serveur d'imprimante…

ouvrir le tunnel comme suit :

```
ssh -N -f -l votre_login -L 9631:printserver.lmpa:631 piccolo.univ-littoral.fr
```

Qu'ès aquò? La commande crée un tunnel ssh (port standard 22 - donc non indiqué - du serveur `piccolo`) en se connectant sur le service d'impression de `printserver` (service `cups` dont le port standard est 631, inaccessible de l'extérieur). Le service `cups` (port 631) est redirigé sur le port 9631 du `localhost` de votre portable. Tout se passe comme si le serveur d'impression du labo était transféré sur l'adresse 127.0.0.1 :9631 de votre portable.

autres commentaires (taper la commande `man ssh` pour plus d'information) :

- `-N` : simple redirection de port
- `-f` : demande l'execution en arrière-plan.
- `-l` votre_login : votre login de connexion sur piccolo (login labo)
- `localhost` : toute machine possède une adresse réseau particulière appelée `loopback`, dont l'adresse IP est 127.0.0.1 (taper la commande `ping localhost` pour vous en convaincre).
- 9631 : le port de redirection peut-être choisi à votre convenance (1234 aurait pu faire l'affaire). Il ne faut pas prendre un port déjà existant (conflit) et un nombre > 1024 (les 1024 premiers ports ne peuvent être ouverts que par l'administrateur)

Ouaip ? mais comment j'imprime ?

Toutes les files d'impression sont désormais accessibles depuis votre `localhost`. Prenons l'exemple la file `hp2` (HP LaserJet M606M recto-verso)

```
# interroger sa disponibilité
lpq -U votre_login -h 127.0.0.1:9631 -Php2
# imprimer le fichier.pdf
lpr -U votre_login -H 127.0.0.1:9631 -Php2 fichier.pdf
```

3.2 Exemple 2 (tunnel) : accéder au serveur de jeton MATLAB

Vous n'êtes pas sur le réseau du laboratoire et vous voulez utilisez Matlab : le service est masqué derrière le pare-feu.

Note : pour plus d'informations sur l'installation et le choix du serveur de jeton , consulter la fiche matlab sur calculco

Le tunnel s'ouvre comme suit :

```
ssh -f -N -L 27000:jetons-cs:27000 -L 39555:jetons-cs:39555 votre_login@calculco.univ-littoral.fr
```

3.3 Exemple 3 (rebond) : accès direct à votre poste de travail

Vous n'êtes pas sur le réseau du laboratoire et vous voulez accéder (échanges de fichiers, lancer un calcul) à votre PC fixe, sans effectuer la manipulation en deux temps (portable serveur PC) ... plus l'éventuel retour !.(cf. fig 3)

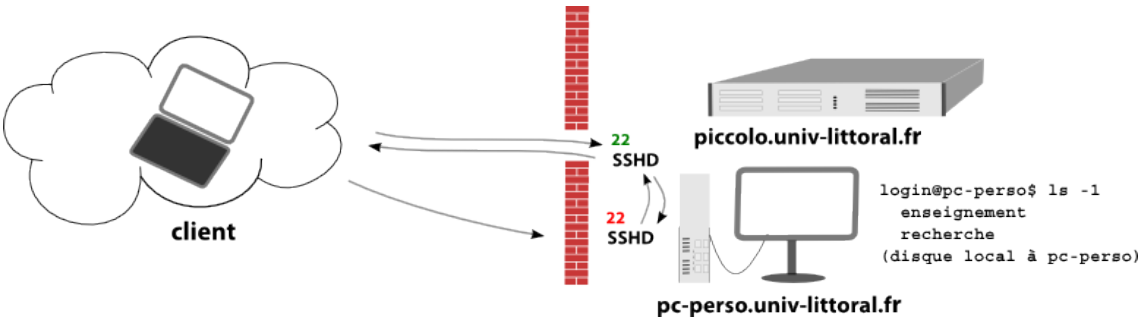

FIGURE 3 – pc fixe

```
ssh -A -t zeblouze@piccolo.univ-littoral.fr ssh -A -t zeblouse@pc-perso.univ-littoral.fr
```

Sauter vite à la section adapter votre fichier config pour simplifier

3.4 Exemple 4 (double rebonds)

On peut, sur le principe de l'exemple précédent, enchaîner plusieurs connexion. S'il s'agissait de se connecter depuis l'extérieur, sur une machine virtuelle (vm-cible) prise en charge par un hyperviseur masqué, on pourrait procéder comme suit :

```
ssh -A -t login1@passerelle ssh -A -t login2@hyperviseur ssh -A login3@vm-cible
```

3.5 Exemple 5 : connexion au serveur « charges des services »

… ou comment accéder à un serveur web « vu depuis » (masqué par firewall, *)? Le serveur des charges n'est accessible que de l'intranet (et encore, cablé : la connexion via eduroam est non-autorisée)

- — avoir accès à une machine du réseau
- — installer un plugin/addons sur votre brouteur (chrome, firefox...)
- — (*)permet aussi d'accéder -par exemple- à sciencedirect (mais « vu de l'ULCO » donc avec les abonnements de l'ULCO)

ouvrir la *socket* : Linux, MacOS, windows 10 WSL :

```
ssh zeblouse@piccolo.univ-littoral.fr -D 12345
```

Note : **12345** est ici arbitraire, le tout est :

- — d'être cohérent (avec la configuration du *proxy* dans le browser - prochaine étape -)
- — de choisir un nombre supérieur à 1024 (les 1024 premiers ports sont réservés à *root* = super-utilisateur)

Avec Putty :
Aller directement dans le menu *SSH/tunnels* (cf. fig 4) :
puis remonter dans le menu *Sessions* pour indiquer le serveur de connexion (`piccolo`) (cf. fig 5) et sauvegarder la session de sorte qu'à la prochaine connexion, il suffira de double-cliquer sur le nom choisi :

configuration du browser : chrome, firefox

- — chrome : « Proxy Switcher and Manager »
- — firefox : « toggle proxy » [lien](https ://addons.mozilla.org/fr/thunderbird/addon/toggle-proxy-51740/)

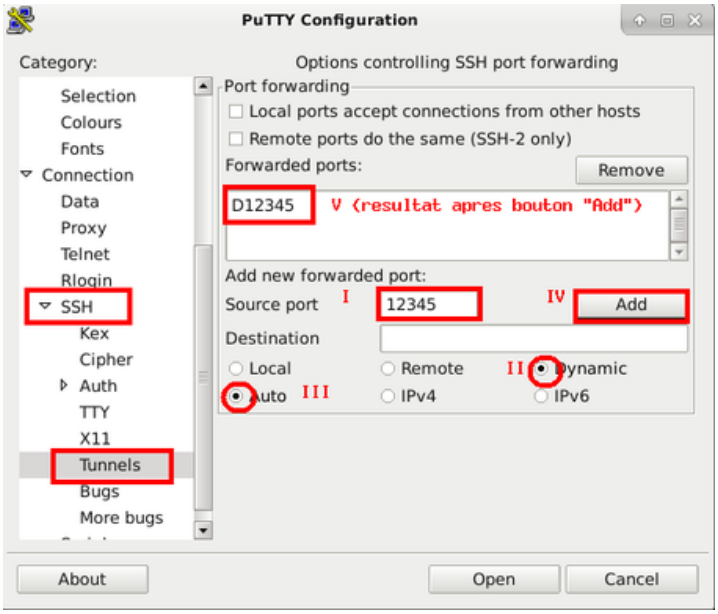

FIGURE 4 – proxy (accès serveur charge enseignement) : configuration de putty

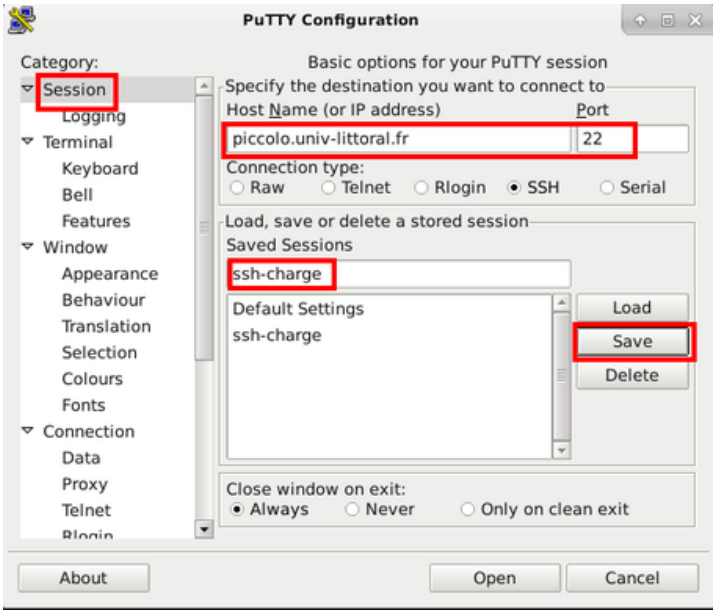

FIGURE 5 – proxy (accès serveur charge enseignement) : configuration de putty

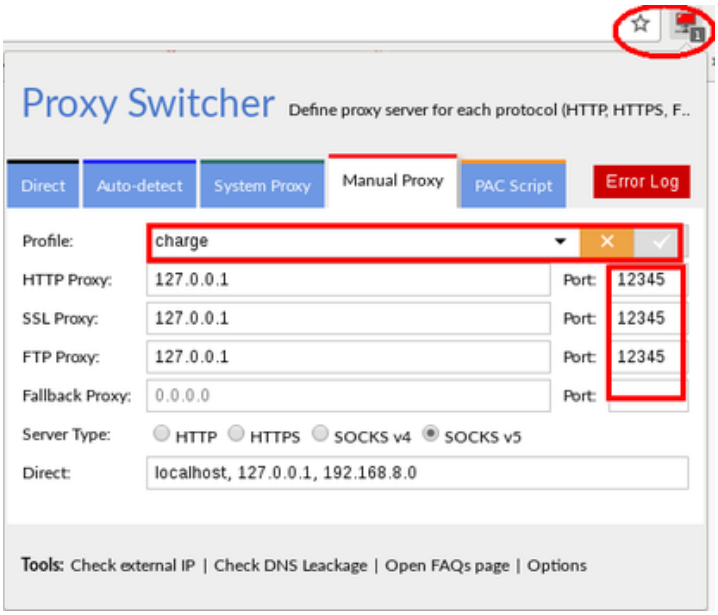

FIGURE 6 – configuration de proxy-switcher : mode manuel

FIGURE 7 – connexion au serveur de gestion des charges enseignant

Une fois le plugin installé, le configurer avec le même numéro de socket que celui choisi lors de la connexion *ssh* (cf. fig 6) :

se connecter au service (cf. fig 7) :

Pour récupérer une connexion « normale », ne pas oublier de switcher cliquer sur *l'applet* en haut à droite de l'écran pour passer d'un mode à l'autre (cf. fig 7) :

— icone rouge : mode proxy

— icone verte : mode normal, sans proxy.

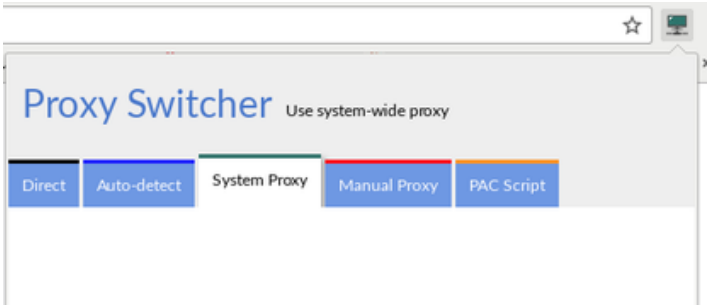

FIGURE 8 – proxy-switcher : repasser en mode « normal » (sans proxy)

Les captures écrans pour le plugin « Toggle Proxy » de firefox (cf. fig 9 et fig 10) :

idem que pour chrome, un bouton à côté de la barre d'adresse permet de passer d'un mode à l'autre (cf. fig 11) :

4 Simplifier le tout(!)

Pour mettre en oeuvre les 4 exemples précédents, éditer (ou créer) le fichier `~/.ssh/config` du portable :

4.1 proposition de fichier `.ssh/config`

proposition de fichier .ssh/config

```
# acces direct sur les serveurs accessible de partout
# il s'agit ici d'un simple alias
Host piccolo
  Hostname piccolo.univ-littoral.fr

# agathe fait le choix d'une clef spécifique pour calculco (clef "calcul")
# elle a un 3ième login : azeblouse
Host calculco
  Hostname calculco.univ-littoral.fr
  IdentifyFile /home/agathe/.ssh/calcul
  User azeblouse
```

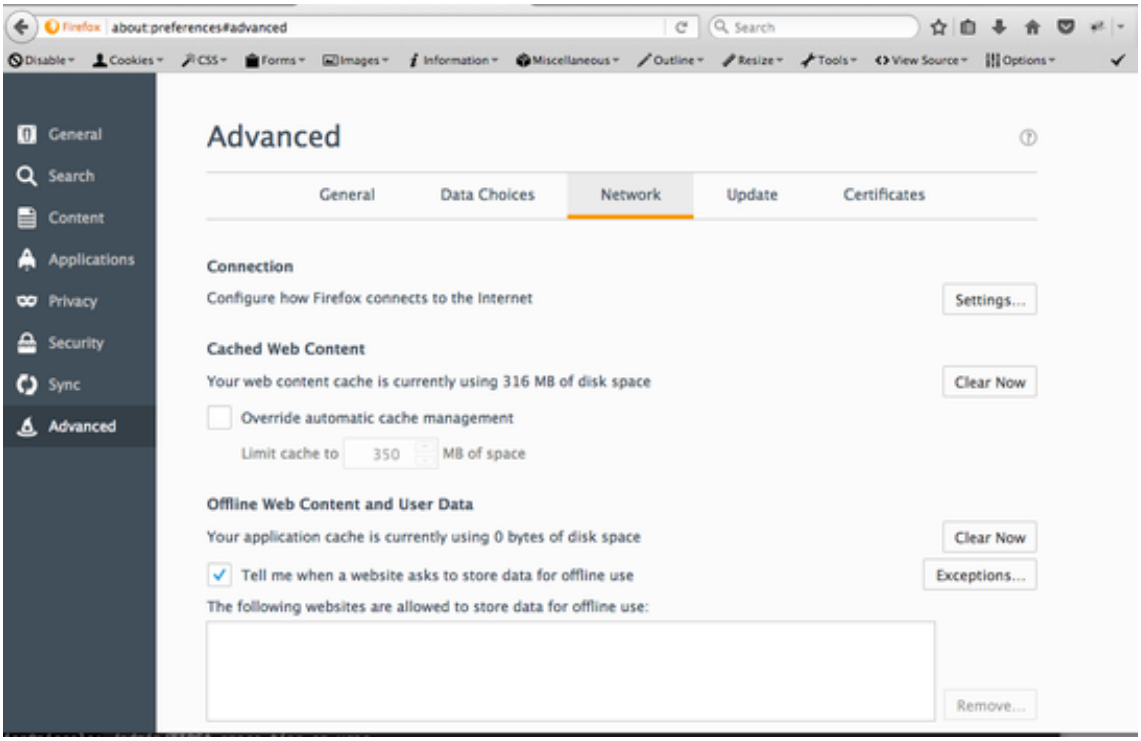

FIGURE 9 – Firefox (plugin « Toggle-Proxy » : network parameters

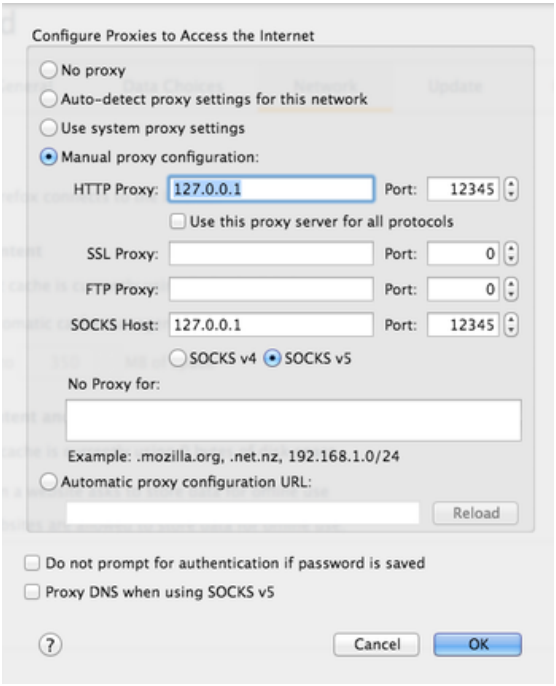

FIGURE 10 – Firefox (plugin « Toggle-Proxy » : proxy parameters

FIGURE 11 – Firefox, « Toggle-Proxy » : switcher mode normal/ mode proxy
:

```
# billard "une bande" (acces pc-perso ...)
Host pc-perso
  Hostname pc-perso.lmpa
  ProxyCommand ssh -W %h:%p piccolo
  User zeblouze

# billard "deux bandes"
# exemple: acces a une machine virtuelle (ma-vm) situee sur un hyper-
# viseur, (ici flute) inaccessible directement de l'exterieur.

Host fluteproxy
  Hostname flute.univ-littoral.fr
  ProxyCommand ssh -W %h:%p piccolo
  User zeblouze

Host ma-vm
  Hostname ma-vm
  ProxyCommand ssh -W %h:%p fluteproxy
  IdentifyFile /home/agathe/.ssh/virtual
  User autre_login


### TUNNELS ###

# matlab du Pôle Calcul
Host matlab
  Hostname calculo
  LocalForward 27000 jetons-cs:27000
  LocalForward 39555 jetons-cs:39555
# (note: ici, calculco, sans l'extension univ-littoral.fr,
# reprend implicitement l'entrée \og{}calculco\fg{} définie plus haut )

# pour imprimer depuis eduroam
Host print
  Hostname piccolo.univ-littoral.fr
  LocalForward 9631 printserver.lmpa:631
  User zeblouse
```

4.2 se connecter, échanger des fichiers

Une fois le fichier `~/.ssh/config` renseigné comme précédemment indiqué, Agathe peut taper les commandes suivantes, de façon homogène et simplifiée, sans mot de passe, sans se soucier ni de ses identifiants ni du nombre de sauts à effectuer pour atteindre la machine *cible* avec laquelle elle compte travailler :

```
# connexions
ssh piccolo
ssh calculco
ssh ma-vm

# copies recursive du répertoire "code source" dans le
# répertoire "projet" du serveur de calcul
scp -rp code-source calculco:projet

# recuperer les resultats
scp  calculco:projet/results/.out .

# copie d'un fichier sur le
scp examen1.tex pc-perso:Licence/

# synchro du rép. recherche local avec celui du serveur de fichiers
rsync -avz recherche/ piccolo:recherche/
...
```

4.3 proxy web (serveur des charges)

```
ssh piccolo -D 12345
```

puis ouvrir le brouteur (cf exemple 5)

4.4 tunnels

A partir de là les commandes pour ouvrir les tunnels (impression et matlab) se réduisent à :

```
# acces imprimantes
ssh -f -N print
# acces à Matlab
ssh -f -N matlab
```

Un petit plus : éditer le fichier `~/.bashrc` du portable Mac ou Linux

```
# établir les tunnels pour l'impression et accès aux jetons matlab
 tprint='ssh -f -N print'
 tmatlab='ssh -f -N matlab'


# alias pour imprimer (une fois le tunnel établi !)

# impression recto-verso sur HP 606M et HP 608M (salle commune)
 imp2='lpr -U votre_login -H 127.0.0.1:9631 $1 -Php2'
 imp='lpr -U votre_login -H 127.0.0.1:9631 $1 -Php'

#interrogation hp hp2
 impq2='lpq -U votre_login -h 127.0.0.1:9631 -Php2'
 impq='lpq -U votre_login -h 127.0.0.1:9631 -Php'
```

Ainsi, au final, les commandes se résument à :

```
# avant de lancer matlab, ouvrir le tunnel
tmatlab

# ouverture du tunnel pour les impressions:
tprint

# imprimer un fichier (recto-verso, sur HP606M)
imp2 le_fichier.ps
```

astuce : complétion pour MacOS

Les fainéants (comme moi) qui veulent ajouter la *completion* sur les *Host* définies dans le fichier de `.ssh/config` peuvent suivre les instructions : gist.github.com/aliang/1024466. Ceci est inutile pour linux (la fonctionnalité est en principe incluse dans votre shell).

Quelques liens/sources :

- formation-debian.via.ecp.fr/ssh.html
- plm-doc.math.cnrs.fr/doc/spip.php?article39
- www.tuteurs.ens.fr/internet/loin/tunnel.html