Le tutoriel dont vous êtes l'héroïne ou le héros : Unix, SSH et le travail à distance

—

Ensimag 1A

Grégory Mounié

2022-2023

Avant de commencer ce tutoriel

Ce tutoriel s'adresse à ceux ayant fini la lecture, et les 80 exercices, du *Guide* et terminé le premier jeu de piste (il y en a deux !). Il propose quelques aspects supplémentaires sur le travail à distance avec SSH et ses amis. Pour rester sur une thématique "à la carte", ce tutoriel peut être suivi de manière linéaire, ou en choisissant ce qui vous intéresse.

Si les indications ne vous suffisent pas, n'hésitez pas à poser des questions à votre encadrant de TP.

1. Tutoriel SSH sur les machines de l'Ensimag

L'une des forces d'UNIX est de pouvoir tout faire dans un terminal shell. Comme il est facile d'obtenir à distance un accès à un terminal avec SSH, il est donc facile travailler sur un ordinateur distant sous UNIX. Nous allons voir quelques aspects autour de SSH, le couteau suisse de votre travail à distance.

Depuis le premier confinement Covid en 2020 aux US, SSH est disponible par défaut aussi dans Windows. Il était déjà disponible depuis sa création au millénaire dernier sous Unix et MacOS X.

Tous les `ensipc` utilisent votre répertoire HOME réseau. Lorsque vous vous connectez sur un autre ensipc, il n'y a de ce fait pas de différence ces différents *montages* de votre

répertoire personnel [1].

Il existe néanmoins un répertoire qui est différent sur chaque machine : le répertoire `/tmp/`. C'est le répertoire où stocker temporairement des fichiers pour les processus tournant sur la machine. Nous l'utiliserons donc pour les exemples qui vont suivre.

Par convention, dans les codes et dans les prompts, nous noterons `ensipcYOURS`, la machine locale sur laquelle vous êtes connectées, ou connectés, et `ensipcTHEIRS` la machine distante. Votre login Ensimag sera noté `LOGIN`.

Entre les machines de l'ensimag, vous n'avez pas besoin de taper votre mot de passe, mais vous devez exporter votre ticket kerberos pour avois accès à votre HOME. Pour cela, il suffit d'ajouter l'option `ssh -K ...`.

Vous pouvez, à partir de ce point :

- Commencer par le début : **obtenir un shell distant**, le partager et savoir le terminer (cf. section 2, page 3).
- **Copier des fichiers entre vos machines** très facilement (cf. section 3, page 6).
- Obtenir **plusieurs shells distants** avec une seule connexion et les laisser tourner même après la fermeture de la connexion (cf. section 4, page 9).
- Manipuler sur votre écran et votre clavier une **application graphique** exécutée à distance (cf. section 5, page 11).
- Pour les plus aventureuses et aventureux, réutiliser une session SSH déjà ouverte (cf. section A, page 12).
- Pour les plus aventureuses et aventureux, faire rebondir votre connexion SSH de machines en machines (cf. section B, page 13).
- Pour les plus aventureux et aventureuses, faire rebondir votre connexion SSH de machines en machines (cf. section C, page 14).
- Pour les plus aventureux et aventureuses, utiliser une machine distante pour accéder au web (cf. section D, page 14).
- Pour les plus aventureuses et aventureux, utiliser une machine distante comme VPN (cf. section E, page 15).
- Comprendre pourquoi ce TP ne présente pas l'utilisation des systèmes de clefs (cf. section 1.1, page 2).
- Découvrir ce qu'il faut changer dans les commandes SSH si vous réalisez ce tp sur votre laptop personnel (cf. section 1.2, page 3).

1.1. Ce que nous ne verrons pas ici : les clefs publiques/clefs privées

Elles servent à se connecter sans avoir à taper votre mot de passe à chaque fois. L'idéal est de les utiliser avec un *agent SSH*.

Ce sujet sera traité plusieurs fois dans votre scolarité, notamment en Réseaux 1A, en TP d'UNIX avancés 1A, pour Gitlab, etc. Nous ne proposons pas de faire, maintenant, des exercices redondants sur ce sujet.

1. Grosse simplification. C'est vrai pour les humains devant les machines, mais pas si vous êtes un ordinateur qui calcule à une vitesse multiple du GHz, comme vous le verrez plus tard en cours de réseaux et de systèmes distribués

1.2. Si vous faites ce TP sur votre laptop personnel

1.2.1. Activer le VPN Ensimag

Sur votre laptop personnel les différences de HOME entre la machine locale et la machine distante sont plus évidentes. Mais pour pouvoir directement faire des connexions SSH avec les ensipc, vous devez avoir activé le VPN de l'Ensimag, ou celui de l'UGA. Les indications de configuration du VPN sont dans le wiki des Bug Busters `https://bugbusters.pages.ensimag.fr/wiki/`

1.2.2. Ajouter votre login aux commandes SSH

Vous n'avez probablement pas le même login entre le compte de votre laptop et celui de l'Ensimag. Il faut alors le préciser dans toutes les commandes SSH.

De même, votre laptop ne considère probablement pas l'Ensimag comme son domaine de référence et donc vous devez spécifier le nom DNS complet des machines.

Dans la suite du sujet, cela signifie qu'à la place de chaque commande du type `ssh -K ensipcTHEIRS ...` vous aurez à taper `ssh LOGIN@ensipcTHEIRS.ensimag.fr ...`

2. Les connexions SSH : obtenir un shell distant

Sous UNIX, quasiment tout peut-être réalisé depuis un shell. L'obtention de ce shell sur une machine distante est le premier pas, et souvent le seul pas dont vous avez besoin.

2.1. Connexion à votre propre machine

Le nom de votre machine est écrit sur une étiquette sur le bord de l'écran. Vous pouvez vérifier en demandant au système avec la commande

```
1 uname -n
```

Connectez-vous à votre machine pour vérifier que son serveur SSH fonctionne.

```
1 you@ensipcYOURS$ ssh -K ensipcYOURS
```

Après avoir donné votre mot de passe, vous devriez vous retrouver dans votre home, en train d'exécuter le même shell. Pas encore très impressionnant !

Deconnectez-vous !

```
1 you@ensipcYOURS$ exit # là, vous êtes dans ssh
2 Connection to localhost closed.
3 you@ensipcYOURS$      # là, vous êtes sortis !
```

2.2. Connexion à la machine de votre voisin

Connectez-vous à la machine d'un de vos voisins : un étudiant ou une étudiante qui fait son TP Unix.

Comme vous pouvez le constater : c'est pareil ! Vous pouvez vous connecter à plusieurs sur la même machine, tant que vous ne lui en demandez pas trop. L'utilisateur connecté devant l'écran n'a pas plus de droit que vous sur la machine [2].

Pour être sûr de la machine exécutant vos commandes et vérifier que votre voisin est bien connecté, vous pouvez taper :

```
you@ensipcYOURS$ ssh ensipcTHEIRS
[...]
you@ensipcTHEIRS$ uname -n
you@ensipcTHEIRS$ who
```

Comparez dans un autre terminal le contenu de votre `/tmp/` avec celui de votre voisin.

```
you@ensipcTHEIRS$ ls -al /tmp
```

et

```
you@ensipcYOURS$ ls -al /tmp
```

Il devrait y avoir quelques différences, en particulier sur le nom des propriétaires de certains répertoires.

Puis deconnectez-vous

2.3. Terminer brutalement une connexion SSH

Il arrive parfois de vouloir demander à SSH de terminer la connexion. Par exemple, si vous avez tué brutalement le serveur SSH distant par exemple en demandant le reboot, ou si une connexion graphique en tâche fond est toujours active.

Le raccourci est de taper «`~.`» en début de ligne. Il faut donc, de temps en temps, les précéder de la touche «Enter» pour passer en début de ligne.

Connectez-vous et interrompez brutalement la connexion.

```
you@ensipcYOURS$ ssh -K ensipcTHEIRS
[...]
you@ensipcTHEIRS$ ~.  # tapez ~. en début de ligne
Connection to localhost closed.  # là, vous êtes sortis !
```

Attention, à n'utiliser uniquement dans les cas nécessaires. Comme pour un kill, c'est un arrêt très brutal. Il a des conséquences tout aussi brutales, comme l'arrêt brusque de vos mauvaises applications qui n'auront pas le temps de sauvegarder vos données.

2. en fait si, quelques-uns. Il a la possession temporaire de l'écran, de la carte son et des ports USB par exemple

C'est à cette perte de données en condition difficile que vous reconnaitrez une mauvaise application (Indication : Vim et Emacs sont vos amis).

2.4. Partager un terminal avec un de vos camarades avec TMATE

Il est relativement aisé de partager un terminal avec un ou une de vos camarades de manière simple. Il suffit de lancer tmate, un fork de tmux qui sera regardé plus en détails à la section 4, page 9.

Dans un terminal lancez `tmate`.

```
you@ensipcYOURS$ tmate
Tip: if you wish to use tmate only for remote access, run: tmate -F
To see the following messages again, run in a tmate session: tmate
↪  show-messages
Press <q> or <ctrl-c> to continue
------------------------------------------------------------------------
Connecting to ssh.tmate.io...
Note: clear your terminal before sharing readonly access
web session read only: https://tmate.io/t/ro-Sz5mtpNaaZgna4feSbQfcW48m
ssh session read only: ssh ro-Sz5mtpNaaZgna4feSbQfcW48m@lon1.tmate.io
web session: https://tmate.io/t/BSSgWxSX5436D7mNStn7VG35v
ssh session: ssh BSSgWxSX5436D7mNStn7VG35v@lon1.tmate.io
```

Ouvrez un autre terminal et lancez dedans la commande de la ligne **ssh session** :

```
you@ensipcYOURS$ ssh BSSgWxSX5436D7mNStn7VG35v@lon1.tmate.io
[...une copie en lecture-écriture du terminal tmate original]
```

Envoyez par email à votre camarade, la commande ssh **en lecture seule (read only)** :

```
ssh ro-Sz5mtpNaaZgna4feSbQfcW48m@lon1.tmate.io
```

Pour avoir une copie interactive de votre terminal, il lui suffit de lancer

```
you@ensipcTHEIRS$ ssh ro-Sz5mtpNaaZgna4feSbQfcW48m@lon1.tmate.io
[... une copie en lecture seule]
```

Continuez dans le terminal tmate, et faîtes quelques actions dedans. Idem dans votre autre terminal. Vérifiez que votre camarade puisse effectivement suivre vos actions avec votre éditeur texte préféré.

```
you@ensipcYOURS$ uname -n
[...]
you@ensipcYOURS$ ls -al
```

```
[...]
you@ensipcYOURS$ votre_editeur_text_préféré (emacs -nw; vim; nano)
```

Vous pouvez quitter le tout !
Vous pouvez, à partir de ce point :

- Revenir aux premières explications où vous trouverez quelques points avancés.
- **Copier des fichiers entre vos machines** très facilement (cf. section 3, page 6).
- Obtenir **plusieurs shells distants** avec une seule connexion et les laisser tourner même après la fermeture de la connexion (cf. section 4, page 9).
- Manipuler sur votre écran et votre clavier une **application graphique** exécutée à distance (cf. section 5, page 11).

3. SSH pour déplacer ses fichiers entre plusieurs machines

3.1. Préparation commune aux autres exercices de la section

Sur les `ensipc`, comme vous avez le même HOME partout, les manipulations de cette partie sont rarement utiles entre deux machines du réseau de l'Ensimag. À l'inverse, elles seront très communes pour transférer des fichiers entre votre laptop et votre compte Ensimag. Notez que le wifi eduroam ou wifi-campus ne fait **pas** partie du réseau de l'Ensimag. Par conséquent, vous aurez besoin du VPN.

Vous connectez par SSH sur la machine de votre voisin. Vous créez un répertoire distant `/tmp/tutossh_LOGIN`.

```
you@ensipcYOURS$ ssh -K ensipcTHEIRS
[...]
you@ensipcTHEIRS$ cd /tmp
you@ensipcTHEIRS$ mkdir Tuto_ssh_LOGIN/
you@ensipcTHEIRS$ cd Tuto_ssh_LOGIN/
you@ensipcTHEIRS$ pwd
/tmp/Tuto_ssh_LOGIN
you@ensipcTHEIRS$
```

3.2. scp

Vous allez créer un fichier `toto.txt` (vide) localement et le copiez dans le répertoire distant (upload). Faites bien cette manipulation depuis la machine locale.

```
you@ensipcYOURS$ touch toto.txt
you@ensipcYOURS$ scp -K toto.txt ensipcTHEIRS:/tmp/Tuto_ssh_LOGIN/
```

Vous vérifiez que le fichier est bien apparu dans le répertoire de la machine distante. Et vous créez un fichier `tata.txt` à distance.

```
you@ensipcTHEIRS$ pwd
/tmp/Tuto_ssh_LOGIN
you@ensipcTHEIRS$ ls
toto.txt
you@ensipcTHEIRS$ touch tata.txt
you@ensipcTHEIRS$ ls
tata.txt   toto.txt
```

Vous copiez le fichier `tata.txt` localement (download).

```
you@ensipcYOURS$ scp -K ensipcTHEIRS:/tmp/Tuto_ssh_LOGIN/tata.txt .
you@ensipcYOURS$ ls
[...] tata.txt   toto.txt [...]
```

3.3. SFTP pour manipulation de fichiers

Il existe un protocole de manipulation de fichiers à distance (FTP). Il tombe en désuétude, car il est complexe à sécuriser. SSH en propose sa propre version améliorée et sûre.

Il est possible de faire quasiment les mêmes manipulations uniquement dans l'interface de sftp. Avec la complétion des noms (TAB) sur les fichiers distants et locaux en prime.

```
you@ensipcYOURS$ sftp -K ensipcTHEIRS
Connected to ensipcTHEIRS.ensimag.fr.
sftp> cd /tmp # cd distant
sftp> mkdir /tmp/Toto_sftp_LOGIN # création distante d'un répertoire
sftp> cd /tmp/Toto_sftp_LOGIN
sftp> put toto.txt # upload
sftp> get toto.txt toto2.txt # download de toto.txt distant dans
↪  toto2.txt local
```

3.4. SSHFS pour la manipulation (presque) transparente de fichiers

Vous pouvez «monter» un répertoire distant, comme vous monteriez le contenu d'une clef USB. Vous pouvez écrire lire ou écrire dedans comme si le répertoire était local. Votre ensipc le fait déjà avec le protocole NFS pour votre HOME. En utilisant sshfs, il est possible de faire la même chose. Il ne faut cependant pas oublier que le réseau devient indispensable. Il est très facile de mettre son laptop en veille en oubliant de «démonter» proprement le répertoire distant. Et tout comme pour une clef USB, ce n'est pas une bonne idée.

Vous montez le répertoire `/tmp/` de l'ensipc votre voisin dans un répertoire vide que vous avez préalablement créé. Vous pouvez également y créer, copier ou lire des fichiers avec votre gestionnaire de fichier graphique préféré.

L'option `-o ServerAliveInterval=15` permet de conserver la connexion active, même si vous ne faites rien et garantie que sshfs se rendra compte au maximum au bout de 45 secondes que la connexion a été perdue.

```
you@ensipcYOURS$ mkdir le_tmp_de_ensipcTHEIRS
you@ensipcYOURS$ sshfs -K -o ServerAliveInterval=15 ensipcTHEIRS:/tmp/
 ↪  le_tmp_de_ensipcTHEIRS
you@ensipcYOURS$ ls le_tmp_de_ensipcTHEIRS
[...] # contenu du /tmp/ de votre voisin
you@ensipcYOURS$ fusermount3 -u le_tmp_de_ensipcTHEIRS
```

3.5. SSH pour l'accès aux fichiers distants par d'autres applications

Les gestionnaires de fichiers comme `Nautilus`, `Thunar`, `Dolphin`, ou `Emacs`, vous permettent d'accéder à vos fichiers distants. Déplacer un fichier entre deux machines se fait alors juste avec un *glisser-déposer*, ou l'équivalent de l'application.

Configurez un accès dans votre gestionnaire préféré avec le `/tmp/` de votre voisin. Pour «nautilus», le file manager de GNOME et par défaut de l'Ubuntu de l'Ensimag, il suffit d'ajouter dans un «nouvel emplacement» un nouveau serveur `ssh://ensipcTHEIRS`. Il est souvent possible de le faire directement avec la ligne de commande.

Par contre, le «démontage» dans nautilus doit être réalisé depuis l'interface graphique, comme pour une clef USB.

Dans l'exemple avec Emacs, la notation de l'URI est celle de son module d'accès à distance TRAMP. Elle fonctionne partout. Emacs ne fait pas vraiment de montage, donc n'a pas besoin de faire un démontage.

```
you@ensipcYOURS$ nautilus ssh://ensipcTHEIRS/tmp/
you@ensipcYOURS$ emacs /ssh:ensipcTHEIRS:/tmp/
```

Ce genre de technique fonctionne aussi avec d'autres services, comme votre Google-Drive par exemple.

Vous pouvez, à partir de ce point :

— Revenir aux premières explications où vous trouverez quelques points avancés.

— Commencer par le début : **obtenir un shell distant**, le partager et savoir le terminer (cf. section 2, page 3).

— Obtenir **plusieurs shells distants** avec une seule connexion et les laisser tourner même après la fermeture de la connexion (cf. section 4, page 9).

— Manipuler sur votre écran et votre clavier une **application graphique** exécutée à distance (cf. section 5, page 11).

4. Screen et Tmux

Parfois, lorsque l'on travaille à distance sur une machine, on peut avoir besoin de couper la connexion avec cette machine (le TGV arrive en gare et vous allez perdre son wifi, la batterie de votre ordinateur portable est presque vide, vous êtes en train de vous endormir sur votre clavier, etc.). La loi de Murphy impose que cela arrive toujours au mauvais moment.

Vous aimeriez aussi pouvoir lancer facilement à distance un autre shell pendant que votre commande préférée s'exécute. Mais vous ne voulez pas lancer une nouvelle connexion SSH.

Il existe deux outils très similaires pour vous aider `screen` et `tmux`. Dans le reste de l'exercice, nous allons utiliser `tmux` qui est plus explicite dans ses commandes.

4.1. Connexion à distance et lancement de tmux

Connectez-vous à pcserveur et lancez la commande `tmux`.

4.2. Lancement du compte à rebours

Lancez un petit compte à rebours de 5 minutes, qui affiche le temps restant toutes les 5 secondes [3].

```
bash -c 'TICS=300; while [ $TICS -gt 0 ]; do printf "$TICS ... "; sleep
↪  5; TICS=$(( TICS - 5 )); done'
```

4.3. Les commandes de tmux

Toutes les commandes de `tmux` commencent par `C-b` (Control b). Si vous connaissez le nom de la commande mais pas son raccourci, vous pouvez taper `C-b :` (Control b puis deux-points, ce qui déclenche la commande *command-prompt*) auquelle vous donnez le nom de la commande. La liste des commandes, et des raccourcis, est disponible avec `C-b ?` (list-keys).

4.4. Détachement et attachement

4.4.1. Détachement et fermeture de la connexion SSH

Tapez la commande `C-b d` (detach-client) pour revenir dans le shell qui a lancé tmux, puis quittez la session SSH en tapant :

```
exit
```

Reconnectez-vous avec SSH, puis lancez la commande

3. `bash -c` sert uniquement si vous avez choisi un autre shell que BASH par défaut (zsh ou tcsh)

```
tmux attach
```

Vérifiez que le compte à rebours est toujours en train de tourner.

4.4.2. Remonter dans l'historique

Remontez dans l'historique en tapant `C-b [` (copy-mode) puis `q` pour terminer. Vous pouvez copier-coller le texte avec les raccourcis Emacs ou vim.

4.5. Ouvrir plusieurs shells

Pour obtenir plusieurs shells s'exécutant simultanément, il existe deux méthodes complémentaires utiles.

4.5.1. Ouvrir une nouvelle fenêtre (window)

Créez un nouveau shell avec la commande `C-b c` (new-window). Navigez entre les shells avec `C-b n` (next-window) et `C-b p` (previous-window).

Naviguez entre les shells 0 et 1 avec les commandes `C-b 1` (select-window) et `C-b 0`.

4.5.2. Scinder une fenêtre horizontalement et verticalement

Créez un nouveau shell en scindant verticalement la fenêtre avec la commande `C-b "` (split-window).

Lancez un compte à rebours dedans.

Créez un nouveau shell en scindant horizontalement la fenêtre avec la commande `C-b %` (split-window -h).

Lancez une horloge dedans avec `C-b t` (clock-mode)

Vous avez maintenant 3 shells dans cette fenêtre. Vous pouvez naviguez entre ces shells avec `C-b o` (select-pane)

4.6. Pour terminer vos différents terminaux

Fermez les différents shells avec la commande `exit` ou `C-d` ou `q` pour le compte à rebours.

Vous pouvez, à partir de ce point :

— Revenir aux premières explications où vous trouverez quelques points avancés.

— Commencer par le début : **obtenir un shell distant**, le partager et savoir le terminer (cf. section 2, page 3).

— **Copier des fichiers entre vos machines** très facilement (cf. section 3, page 6).

— Manipuler sur votre écran et votre clavier une **application graphique** exécutée à distance (cf. section 5, page 11).

5. SSH et les applications graphiques

Nous allons lancer une application graphique, comme `matlab`. Elle est relativement gourmande en ressource graphique. Si vous faites ce tutoriel depuis un réseau à faible débit et grande latence, vous pouvez utiliser des applications moins graphiquement gourmandes, mais graphique quand même, comme `emacs`.

5.1. SSH -XC

Lancez matlab sur le pc de votre voisin et vous l'affichez sur le votre. L'option `X` demande le transport du flux X11. L'option `C` permet de compresser ce flux pour gagner en vitesse.

Puis quittez matlab. Cela devrait interrompre la connexion.

```
1 you@ensipcYOURS$ ssh -KXC ensipcTHEIRS matlab
```

5.2. La fin de la première partie du jeu de piste est ici !

Bravo à vous si vous lisez cette section parce que vous êtes en train de terminer la première partie du jeu de piste !

Si vous en voulez plus, il y a une deuxième partie, beaucoup moins guidée.

5.3. Session X11 distante et détachée (à la TMUX)

Il existe des protocoles modernes qui optimisent plus que l'option `-C` la compression du flux X11 pour gagner encore en vitesse et réactivité.

Les trois outils suivants sont différents dans leurs implémentations, leurs interfaces et leurs philosophies. Ils permettent aussi de travailler de manière détachée, c'est-à-dire de laisser votre application ou votre session tourner à distance alors que vous avez éteint votre machine locale. Un peu ce que fait TMUX dans un terminal. Et, comme pour tmux, il n'y a pas de magie : si la machine s'éteint physiquement, votre application sera terminée.

5.4. Xpra

Xpra permet de laisser votre application graphique tourner sur la machine, à la manière de tmux.

```
1 you@ensipcYOURS$ xpra start ssh://ensipcTHEIRS --start=matlab
```

Une petite icone Xpra a dû apparaitre dans votre environnement. En cliquant dans son menu contextuel, déconnectez-vous.

Puis vous pouvez faire `xpra attach` pour retrouver votre application.

```
1 you@ensipcYOURS$ xpra attach
```

Vous pouvez maintenant terminer le serveur Xpra sur la machine de votre voisin en demandant de faire «shutdown session».

5.5. x2go

`x2goclient` vous offre une interface graphique très jolie. Vous devriez réussir à lancer une session sur la machine de votre voisin tout seul.

Il gère aussi le transport du son jusqu'à votre PC.

5.6. VNC

VNC est un protocole plus ancien, un peu moins performant, mais décliné en de nombreuses variantes, dont par exemple celle de l'utiliser depuis un simple navigateur web sur le poste client.

Néanmoins, cette solution web a quelques limitations (sur l'utilisation des numéros de port) et dépasserait le cadre de ce tutoriel.

Vous pouvez, à partir de ce point :

— Revenir aux premières explications où vous trouverez quelques points avancés.

— Commencer par le début : **obtenir un shell distant**, le partager et savoir le terminer (cf. section 2, page 3).

— **Copier des fichiers entre vos machines** très facilement (cf. section 3, page 6).

— Obtenir **plusieurs shells distants** avec une seule connexion et les laisser tourner même après la fermeture de la connexion (cf. section 4, page 9).

A. SSH : Réutiliser une connexion SSH déjà ouverte

Il est possible d'ouvrir une connexion ssh, puis de se servir de cette connexion (connexion maîtresse) pour faire passer d'autres sessions SSH simultanément, par exemple, pour un scp avec la même machine. Pour cela, il faut choisir un (pseudo-)fichier qui servira aux connexions (le `ControlPath`) et qui ne soit pas accessibles aux autres utilisateurs de votre machine.

Connectez-vous à `ensipcTHEIRS`, en activant le mode Master et en indiquant le fichier (socket) de communication.

```
you@ensipcYOURS$ ssh -K -M
↪  "-oControlPath=~/.ssh/controlmaster-socket/%r@%h:%p" ensipcTHEIRS
```

Dans le répertoire `/tmp/` de `ensipcTHEIRS`, créez un fichier vide à votre nom.

```
cd /tmp/
touch LOGIN
```

Tant que vous ne fermez pas la connexion master, vous pouvez vous connecter facilement dans un autre terminal, sans retaper vos mots de passe.

Dans un nouveau terminal, connectez-vous sans taper vos mots de passe :

```
you@ensipcYOURS$ ssh -K
↪ "-oControlPath=~/.ssh/controlmaster-socket/%r@%h:%p" ensipcTHEIRS
```

Dans un nouveau terminal, copiez le fichier que vous avez créé dans le `/tmp_data` de pcserveur et copiez-le chez vous.

```
you@ensipcYOURS$ scp
↪ "-oControlPath=~/.ssh/controlmaster-socket/%r@%h:%p"
↪ ensiPCTHEIRS:/tmp/LOGIN .
```

Éditez le fichier copié pour ajouter quelques caractères et recopiez-le sur pcserveur pour écraser l'ancienne version.

Dans la connexion master, affichez le contenu du fichier et vérifiez qu'il s'agit bien de la nouvelle version.

Puis effacez le fichier de `ensipcTHEIRS`.

Pour fermer toutes les connexions, il suffirait de fermer la connexion master. Néanmoins, il est aussi possible de le faire depuis une connexion esclave :

```
you@ensipcYOURS$ ssh -K
↪ "-oControlPath=~/.ssh/controlmaster-socket/%r@%h:%p" -O exit
↪ ensipcTHEIRS
```

Vérifiez que les deux connexions ouvertes ont bien été fermées.

B. SSH avec des rebonds sur un bastion

Avoir toutes ses machines directement accessibles depuis Internet demande de constamment sécuriser l'ensemble du parc informatique, pour chaque logiciel installé sur chaque machine. Ne pas avoir d'accès du tout depuis Internet vers ses machines n'est pas très pratique.

Le compromis assez commun, utilisé par l'Ensimag, est d'avoir un routeur exposé qui fournit un service de VPN.

Un autre compromis possible est de n'avoir qu'une seule machine (un bastion) accessible avec très peu de logiciels à maintenir, donc beaucoup plus facile à sécuriser. Parfois avec des systèmes très robustes comme OpenBSD. Le bastion sert de porte d'entrée. Il ne sert pas à travailler dessus.

Pour renforcer encore la sécurité, au lieu d'utiliser des mots de passes, il est aussi possible de limiter les connexions au bastion à des connexions avec des clefs.

Nous allons utiliser `ensipcTHEIRS` pour faire un rebond vers `pcserveur.ensimag.fr`.

Connectez à `ensipcTHEIRS`.

```
you@ensipcYOURS$ ssh ensipcTHEIRS
```

Puis deconnectez-vous avec

```
you@ensipcTHEIRS$ exit
```

L'option `-J` de `ssh` vous permet d'indiquer la machine qui sert pour un rebond (Jump) vers une connexion à une autre machine.

Connectez-vous à `pcserveur` en passant par `ensipcTHEIRS`. Vous taperez deux fois votre mot de passe (une fois pour `ensipcTHEIRS`, une fois pour `pcserveur`).

```
you@ensipcYOURS$ ssh -K -J ensipcTHEIRS pcserveur.ensimag.fr
```

Vérifiez que vous êtes bien sur votre compte, sur pcserveur, puis déconnectez-vous avec

```
you@pcserveur$ uname -n
you@pcserveur$ id
you@pcserveur$ who
you@pcserveur$ exit
```

C. La configuration de SSH pour ne pas taper les options

Pour éviter de toujours avoir à taper les options qui vous intéressent avec une machine particulière, vous pouvez ajouter cette option dans votre `~/.ssh/config`.

Par exemple, l'option `ProxyJump` pour vous connecter à `pcserveur` depuis ensipcTHEIRS :

```
Host pcserveurViaTHEIRS # un nom arbitraire, typiquement "pcserveur"
User LOGIN
HostName pcserveur.ensimag.fr
ProxyJump LOGIN@ensipcTHEIRS
```

et il vous suffira alors de taper

```
ssh -K pcserveurViaTHEIRS
```

D. SSH comme proxy web

La commande `ssh` permet d'utiliser un serveur `ssh` comme serveur mandataire (proxy) qui permet de surfer depuis le serveur mandataire. Par exemple, en utilisant `ensipcTHEIRS` il devient possible d'accéder aux serveurs web de l'école comme si vous étiez sur ensipcTHEIRS.

Utilisez `ensipcTHEIRS` comme machine mandataire pour surfer :

```
ssh -D localhost:5888 ensipcTHEIRS
```

Modifiez les préférences de votre navigateur pour indiquer qu'un serveur SOCKS est disponible localement (localhost) au port 5888. (Dans Firefox : Preferences, General, Network Settings)

Surfez sur l'intranet de l'Ensimag.

Pensez à terminer la connexion et à remettre la configuration de votre navigateur en accès direct vers Internet.

E. SSH comme VPN

Il est possible d'utiliser SSH pour mettre en place un VPN, mais cela demande des droits d'administrateurs sur la machine hébergeant le serveur `SSH`. Cela vous servira peut-être sur vos propres machines, mais vous ne pouvez pas le mettre en place à l'Ensimag.

Pour en savoir plus, nous vous conseillons de lire les différents manuels de SSH, très complets et avec de nombreux exemples

```
you@ensipcYOURS$ man ssh
you@ensipcYOURS$ man ssh_config
you@ensipcYOURS$ man ssh-agent
you@ensipcYOURS$ man ssh-add
```