université de BORDEAUX

TP0 : Environnement de Travail du CREMI

1 Télétravail au CREMI

Voici deux ressources utiles pour apprendre à travailler à distance :

— https://services.emi.u-bordeaux.fr/intranet/spip.php?article175

— https://gforgeron.gitlab.io/ssh/guide.pdf

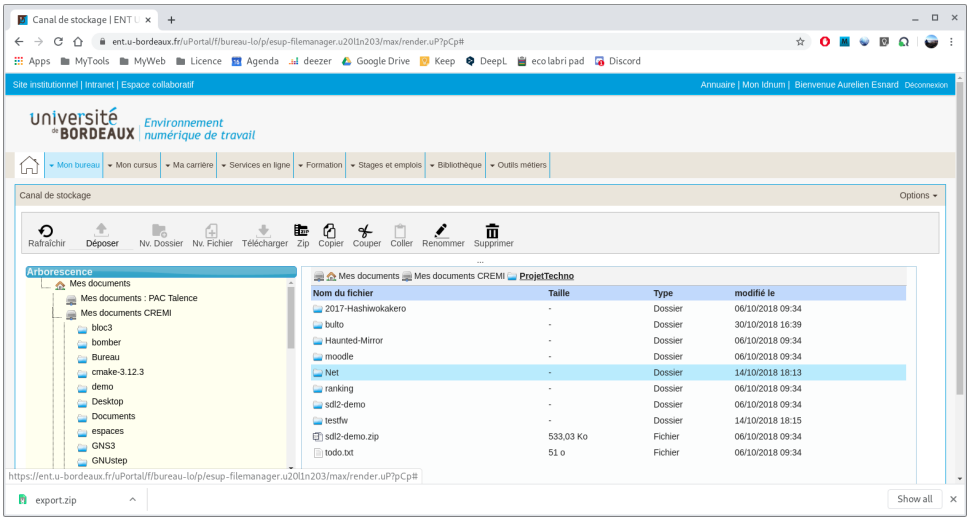
Il nous semble important que vous puissiez faire les deux choses suivantes afin d'arriver à travailler correctement depuis chez vous, ce TP reprend la plupart des éléments décrits dans ces documentations :

1. échanger des fichiers entre votre ordinateur personnel et votre compte au CREMI ;
2. vous connecter aux machines du CREMI pour continuer à travailler à distance : exécuter des commandes en ligne, éditer des fichiers, développer des programmes (C, Python, ...), les compiler et les exécuter, ...

Il existe de multiples solutions pour réaliser ces objectifs, dépendant aussi de votre système d'exploitation : Linux, Windows, ... En outre, il faudra toujours privilégier (lorsque c'est possible) les outils qui fonctionnent en mode texte en comparaison aux outils graphiques plus gourmands en ressources !
Nota Bene : Pour accéder au CREMI, l'authentification est nécessaire avec votre login et mot de passe habituel.

1.1 Échange de fichiers avec le CREMI

Voici un premier exercice facile qui nécessite juste l'utilisation d'un navigateur Web... Commencez par vous connecter à l'ENT de l'Université de Bordeaux : https://ent.u-bordeaux.fr/. Puis sélectionnez le menu `Bureau > Canal de Stockage`... Patientez quelques secondes et vous devriez avoir accès au fichiers de votre compte informatique du CREMI.



Vous pouvez naviguer dans l'arborescence de vos fichiers, télécharger un fichier (bouton Télécharger) ou un répertoire entier sous forme d'une archive (bouton Zip) et vous pouvez également créer de nouveau répertoire (bouton Nv. Dossier), déplacer des fichiers (bouton Copier/Couper/Coller), ou encore déposer un ou plusieurs fichiers de votre machine vers le CREMI (bouton Déposer), etc.
Exercice : Faites un essai de téléchargement d'un fichier (puis d'un répertoire) du CREMI vers votre machine et d'envoi d'un fichier de votre machine vers le CREMI.

1.2 Connexion SSH au CREMI

Pour travailler à distance en utilisant les machines du CREMI, vous pouvez utiliser SSH directement depuis une machine Linux ou MacOS.
A propos de Windows : Si vous n'avez qu'un système Windows, il vous faut installer un client SSH.

- À partir de Windows 10, il suffit d'installer le composant OpenSSH, il se trouve dans "Applications et fonctionnalités", "Gérer les fonctionnalités facultatives", "OpenSSH Client". Cela suffira pour les parties non graphiques.
- Pour les parties graphiques (ou si vous ne pouvez pas installer OpenSSH, on peut utiliser MobaXterm et cliquer sur le bouton "Start local terminal", cela vous lance un shell où vous pourrez lancer toutes les commandes documentées ci-dessous.
- vous pouvez aussi utiliser WSL (Windows Subsystem for Linux) qui permet d'intégrer par exemple un Linux/Ubuntu dans Windows 10.

Attention, chacune de ces 3 options vit dans son propre monde, si vous en utilisez plusieurs, il faut suivre ce tutoriel pour chaque pour générer les clés. Si vous utilisez VSCode, il faut utiliser le composant OpenSSH pour pouvoir éditer facilement les fichiers à distance.

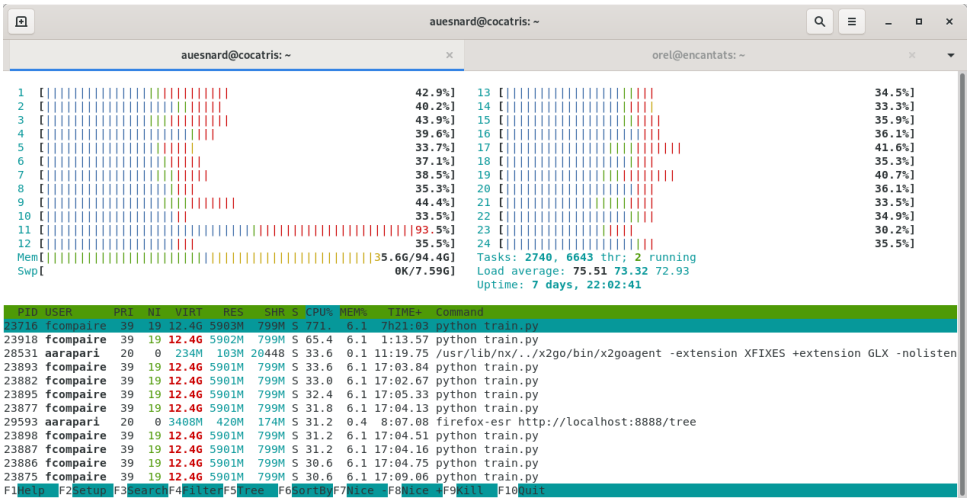
La première étape consiste à se connecter à la passerelle du CREMI qui est la machine *jaguar.emi.u-bordeaux.fr*, le seul point d'entrée et de sortie du CREMI depuis/vers Internet. Commençons donc par nous connecter à la passerelle *jaguar.emi.u-bordeaux.fr* avec la commande *ssh* sous Linux, en remplaçant *mylogin* par votre *login* au CREMI. On vous demande votre mot de passe CREMI.

```
toto@myhome$ ssh mylogin@jaguar.emi.u-bordeaux.fr
mylogin@jaguar$
```

Une fois connecté à la passerelle *jaguar*, on doit se connecter en rebond à une autre machine du CREMI, comme par exemple les serveurs *boursouflet, cocatris, infini1, infini2, jolicoeur, mcgonagall, trelawney* ou *xeonphi*. Notons que la passerelle *jaguar* n'est pas faite pour travailler et d'ailleurs il n'y a pas d'outils pour le faire. On recommence avec la commande ssh, mais cette fois-ci il n'est plus utile de préciser son login, qui est déjà bien configuré. Par exemple pour rebondir sur le serveur cocatris :

```
mylogin@jaguar$ ssh cocatris
mylogin@cocatris$
```

Une fois connecté sur un serveur vous pouvez contrôler sa charge (load average), son % CPU et son occupation mémoire rapidement en utilisant la commande w, top ou htop. (Tapez 'q' pour sortir de top ou de htop.) Voici un exemple le résultat de la commande htop sur un serveur surchargé ! Ici la charge de cocatris est de 75, alors qu'il n'y a que 24 "cœurs"... Disons que ça rame, car la charge maximale ne devrait pas dépasser le nombre de cœurs pour un bon fonctionnement !



Que faire si les serveurs sont trop chargés ? Pour choisir choisir une machine libre pour travailler et éventuellement la réveiller à distance, il faut consultez la page suivante dans l'Intranet du CREMI : https://services.emi.u-bordeaux.fr/exam/?page=wol

Par exemple, si je choisis la salle 008 qui a des machines très puissantes (autant que la plupart des serveurs !) dans le lien ci-dessus, je peux démarrer (grâce à la technologie *Wake-on-LAN*) la machine *miro* pour moi tout seul... Et ensuite, je m'y connecte (en n'oubliant pas de me déconnecter de *cocatris* au préalable) :

```
mylogin@cocatris$ exit
mylogin@jaguar$ ssh miro
mylogin@miro$ w
  18:41:49 up 13 min,  1 user,  load average: 0,25, 0,47, 0,50
  USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
  mylogin pts/0    2001:660:6101:80 18:41    2.00s  0.19s  0.00s w
```

Je constate avec la commande *w* que je suis le seul utilisateur et que la charge est inférieure à 1... On va pouvoir commencer à travailler au calme...
Important : lors d'un Wake-on-LAN après 22h il faut impérativement choisir "mode travail de nuit".

1.2.1 Travailler en mode texte...

La connexion SSH ne permet par défaut que de lancer des commandes en mode texte... Ca peut être suffisant dans certains cas, si vous souhaitez éditer des fichiers, compiler et exécuter des programmes... Dans ce cas, si vous êtes débutant, je vous recommande d'utiliser *nano* comme éditeur de texte... Ca dépanne, c'est très minimaliste, mais ça ne permet en aucun cas de développer efficacement un programme ! Les raccourcis sont indiqués en bas de page : `Ctrl-O` par exemple pour sauvegarder un fichier, `Ctrl-X` pour quitter... Il existe des alternatives plus puissantes fonctionnant en mode texte comme *vim* ou *emacs*, mais il faudrait un cours entier pour en parler !

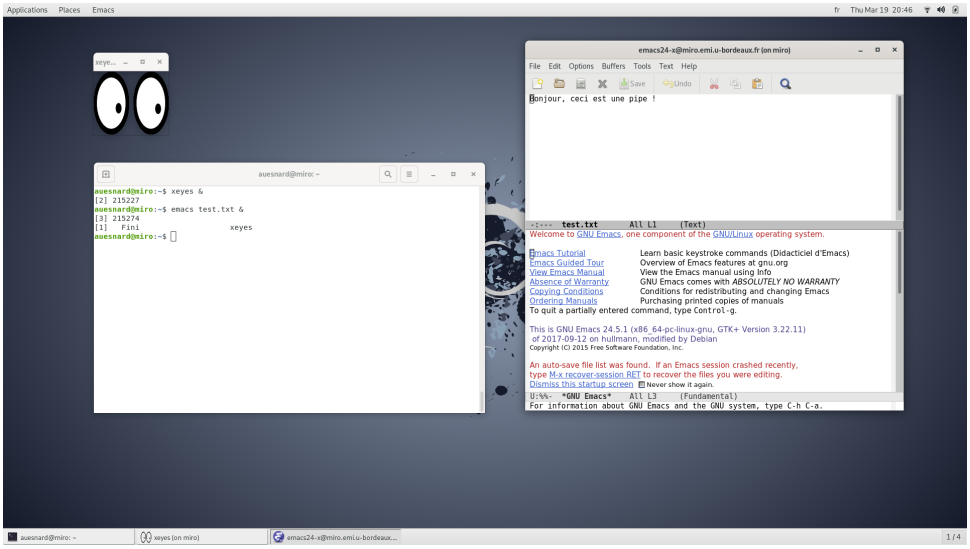
```
mylogin@miro$ nano
```

1.2.2 Et les applications graphiques ?

L'option -X (ou -Y) de la commande *ssh* permet de faire du *Forward X11*, c'est-à-dire de lancer des applications graphiques à distance, mais c'est souvent difficile à utiliser, surtout si la connexion Internet n'est pas bonne ou que le serveur du CREMI est trop chargé ! Pour les utilisateurs Windows, il faut soit utiliser MobaXterm qui intègre un serveur X, soit activer cette option dans *Putty* et installer un serveur X...

```
toto@myhome$ ssh -X mylogin@jaguar.emi.u-bordeaux.fr
mylogin@jaguar$ ssh -X miro
mylogin@miro$ xeyes &
mylogin@miro$ emacs &
mylogin@miro$
```

Et voici le résultat...



Mais ça reste néanmoins lent et peu fonctionnel quand on n'a qu'une ligne ADSL ! De plus, il peut y avoir quelques difficultés avec des applications graphiques plus complexes (OpenGL). Mieux vaut privilégier le mode texte, ou des modes de travail alternatifs... je développe chez moi et je synchronise mes fichiers grâce à un dépôt Git interne comme Savane au CREMI ou GitHub par exemple...

1.2.3 Marre de taper toujours son mot de passe ?

Afin d'utiliser toutes les possibilités de SSH, il est recommandé de créer un couple de clé privé/public RSA (cryptographie asymétrique) qui permet de s'authentifier auprès d'une machine distante sans mot de passe Unix ! En pratique sous Linux, il faut commencer par créer son couple de clés sur sa machine (à la maison) avec la commande *ssh-keygen* (sans *passphrase*[1] juste taper 'entrée' trois fois pour laisser le chemin proposé tel quel et sans passphrase) :

```
toto@myhome$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/mylogin/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/mylogin/.ssh/id_rsa.
Your public key has been saved in /home/mylogin/.ssh/id_rsa.pub.
```

Tous les fichiers importants sont stockés dans le répertoire .ssh/ (caché) à la racine de votre compte : le fichier *id_rsa* contient votre clé privé (qui doit rester absolument secrète) et le fichier *id_rsa.pub* contient la clé publique associée à la clé privée. Afin de

1. La *passphrase* n'est pas un mot de passe qui permet de s'authentifier sur une machine distante, mais un mot de passe qui sert à encrypter le fichier de la clé privé. Il est généralement recommandé d'utiliser une passphrase en association avec un *agent* SSH... Néanmoins, pour simplifier ce tutoriel, nous allons nous en passer.

pouvoir se connecter sur une machine distante avec SSH, il va falloir ajouter notre clé publique dans le fichier *.ssh/authorized_keys* de la machine visée, ce que l'on peut faire simplement en utilisant la commande *ssh-copy-id*.
Tapez les commandes suivantes, toujours en remplaçant *mylogin* par son *login* au CREMI :

```
toto@myhome$ ssh-copy-id mylogin@jaguar.emi.u-bordeaux.fr
```

Note : si vous utilisez le composant OpenSSH sous Windows, `ssh-copy-id` n'est malheureusement pas disponible. Il vous faut alors le faire à la main :

```
toto@myhome$ scp ~/.ssh/id_rsa.pub mylogin@jaguar.emi.u-bordeaux.fr
toto@myhome$ ssh mylogin@jaguar.emi.u-bordeaux.fr
mylogin@jaguar$ cat id_rsa.pub >> .ssh/authorized_keys
```

Attention à bien mettre deux charactères `>` pour ajouter les clés sans supprimer celles que vous auriez déjà mises.

Et vérifiez que vous arriver à vous connecter au CREMI sans mot de passe !

```
toto@myhome$ ssh mylogin@jaguar.emi.u-bordeaux.fr
mylogin@jaguar$
```

Essayez maintenant de rebondir sur un serveur du CREMI, comme par exemple *cocatris*... Ouch, on vous demande à nouveau un mot de passe ! Il va falloir répéter les étapes précédentes sur la machine *jaguar* au CREMI :

```
toto@myhome$ ssh mylogin@jaguar.emi.u-bordeaux.fr
mylogin@jaguar$ ssh-keygen
...
mylogin@jaguar$ ssh-copy-id cocatris
...
mylogin@jaguar$ ssh cocatris
mylogin@cocatris$
```

Youhou, ça marche ! Depuis *jaguar*, essayez de vous connecter (sans mot de passe) à un autre serveur, comme *infini1*... Ca marche tout seul, car toutes les machines du CREMI partage le même fichier *.ssh/authorized_keys* grâce au système de fichiers en réseau (NFS).
Exercice : Reprenez ce tutoriel et vérifiez que vous arrivez à vous connecter à un serveur de votre choix au CREMI (*jolicoeur, cocatris, inifni1, ...)* sans mot de passe.

1.2.4 Fichier de configuration

Un truc à faire pour se faciliter la vie consiste à créer un fichier de configuration `~/.ssh/config` avec le contenu suivant (toujours en remplaçant *mylogin* par son *login* au CREMI) :

```
# options par défaut
ForwardAgent yes
ForwardX11 yes
User mylogin

# gateway
Host cremi
Hostname jaguar.emi.u-bordeaux.fr

# internal server
Host cocatris
ProxyJump cremi
```

Attention, c'est bien dans `~/.ssh/config` et pas ailleurs. Sous Windows avec WSL il vous emmène par défaut ailleurs que dans `~`, il faut donc bien préciser le `~`

Après quoi, on peut taper les commandes suivantes pour se connecter directement aux machines du CREMI déclarées dans le fichier de *config* :

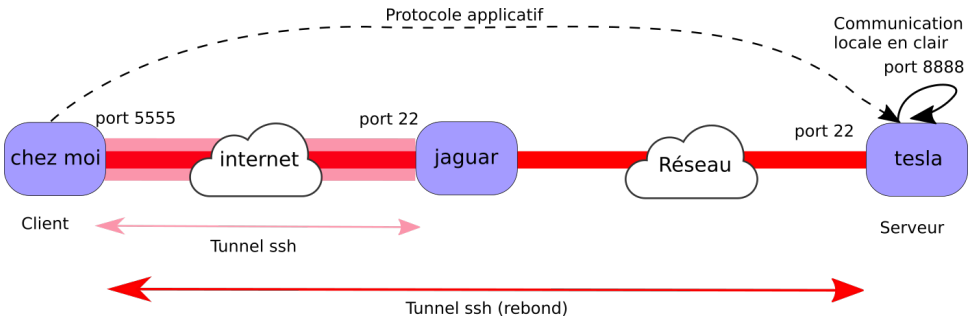
```
# exemple 1: connexion à jaguar
toto@myhome$ ssh cremi
mylogin@jaguar$

# exemple 2 connexion à cocatris
toto@myhome$ ssh cocatris
mylogin@cocatris$
```

En outre, il existe une option `-J` qui permet de *cascader* des connexions SSH, même si la machine *inifini1* n'est pas délarée dans le fichier de *config* :

```
toto@myhome$ ssh -J cremi infini1
mylogin@infini1$
```

1.3 Tunnel SSH (Bonus)

Dans le cas ou vous où vous souhaiteriez accéder depuis chez vous (maison) à un serveur applicatif qui n'est accessible que depuis l'intérieur du CREMI, il est nécessaire de mettre en place ce qu'on appelle un *tunnel*. Le tunnel est composé de deux parties, une partie qui est une communication *SSH* entre votre machine et la passerelle *jaguar* et une autre qui est une communication dans le protocole que vous avez choisi entre *jaguar* et le serveur applicatif ciblé. Un schéma descriptif de ce que l'on veut mettre en place est fourni ci-dessous :



Dans cet exemple, nous souhaitons accéder à une service réseau (disons un serveur *netcat* à l'écoute sur le port 8888) de la machine *tesla* qui n'est accessible que depuis l'intérieur du CREMI. Commençons par nous connecter à la machine tesla avec SSH en utilisant l'option `-J` pour faire un rebond et lançons le serveur *netcat* :

```
toto@myhome$ ssh -J cremi tesla
mylogin@tesla$ netcat -l -p 8888
```

Nota Bene : Si vous n'avez pas de fichier fichier de configuration SSH (cf. section précédente), il faut remplacer dans la commande ci-dessus le nom de machine `cremi` par `mylogin@jaguar.emi.u-bordeaux.fr`.

Pour mettre en place le tunnel SSH, il faut utiliser la commande *ssh* suivante (à exécuter depuis votre machine personnelle) :

```
toto@myhome$ ssh -N -J cremi tesla -L 5555:localhost:8888
```

Cette commande va créer un tunnel jusqu'à la machine *tesla* et va attendre sans rendre la main[2]. Ainsi une connexion sur le port 5555 de votre machine locale (*localhost*) sera transférée par le tunnel SSH à *tesla* qui effectuera à la sortie du tunnel une connexion *en clair* vers le serveur *netcat* de la machine locale (i.e. `tesla:8888`).

Exercice : Vérifiez le bon fonctionnement de ce tunnel. Il suffit pour se connecter d'utiliser une client netcat sur *localhost :5555*, qui est le point d'entrée du tunnel mis en place avec la commande précedente.

```
toto@myhome$ netcat localhost 5555
```

On peut appliquer cette technique dans de multiples contextes afin d'utiliser un service réseau, comme s'il était local à sa machine, alors que celui-ci n'est pas disponible sur Internet !

2. Pour supprimer le tunnel, il faut juste faire un Ctrl-C à l'endroit où le tunnel est lancé... Et c'est tout !