Utilisation du shell Unix
8 - Outils réseaux

1 Connexion confortable à gigondas

Pour vous connecter au serveur `gigondas` depuis un poste Linux, vous pouvez utiliser la commande :

```
ssh -X gigondas
```

L'option `-X` autorise le serveur `gigondas` à ouvrir des applications graphiques dans votre environnement local (par exemple `geany` ou encore `xterm`).

Testez cette commande et ouvrez un terminal depuis `gigondas` (avec la commande `xterm`).

NB : avec les paramètres par défaut, la fenêtre `xterm` n'est pas très lisible, vous pouvez ajouter des options pour un meilleur rendu (police de caractère `Inconsolata`, taille 12, 108 colonnes et 32 lignes) : `xterm -fa Inconsolata -fs 12 -geometry 108x32`

Vous pouvez aussi tester directement :

```
ssh -X gigondas xterm
ssh -X gigondas xterm -fa Inconsolata -fs 12 -fg Green -bg Grey12
```

On aimerait pouvoir ouvrir un terminal sur `gigondas` sans être obligé de saisir à chaque fois le mot de passe. Pour cela nous devons utiliser un *agent* `ssh` qui se chargera de l'authentification pour nous.

Revenez dans une fenêtre du PC Linux de la salle de TP. Puis créez une paire de clés (publique/privée) :

```
ssh-keygen
```

Pour le nom du fichier, gardez la valeur par défaut qui est proposée (`$HOME/.ssh/id_rsa`).

Pour la *passphrase* qui vous est demandée, le plus simple est d'utiliser votre mot de passe de session.

Vous disposez maintenant d'une paire *clé publique*/*clé privée* dans le répertoire `$HOME/.ssh`. Placez-vous dans ce répertoire et vérifiez qu'il contient bien les deux fichiers : `id_rsa.pub` (clé publique) et `id_rsa` (clé privée).

Pour autoriser la connexion à `gigondas` avec cette clé, vous devez installer la clé **publique** dans le fichier `authorized_keys` de votre répertoire `.ssh` sur `gigondas` :

1. Copiez la clé publique (`id_rsa.pub`) sur `gigondas` avec la commande `scp` (*secured copy*) :
   ```
   scp id_rsa.pub gigondas:
   ```
   Attention à ne pas oublier le `':'` après `gigondas` car sinon vous allez créer une copie locale du fichier ;
2. Connectez-vous à `gigondas` ;
3. Placez-vous dans le répertoire `.ssh` ;
4. Ajoutez la clé publique (contenu du fichier `id_rsa.pub`), au fichier `authorized_keys` :
   - si le fichier `authorized_keys` n'existe pas vous pouvez simplement renommer le fichier `id_rsa.pub` :
     ```
     mv ../id_rsa.pub authorized_keys
     ```
   - si `authorized_keys` existe déjà :
     ```
     cat ../id_rsa.pub >> authorized_keys
     ```
5. Déconnectez-vous de `gigondas`.

Note : les manipulations précédentes sont les manipulations « normales » pour installer une clé sur un serveur distant comme `gigondas`, mais à l'IUT on aurait pu grandement les simplifier puisque votre *HOME* est le même sur les postes Linux et sur `gigondas` [1].

Si vous lancez maintenant une connexion à `gigondas` avec la commande `ssh`, vous devriez voir apparaître une fenêtre du type de celle de la figure 1, vous invitant à déverrouiller votre clé privée pour la rendre lisible par l'agent SSH (agent lancé par défaut à l'ouverture de session).

N'hésitez pas à cocher la case Déverrouiller automatiquement cette clé quand je suis connecté.

Si cette fenêtre n'apparaît pas, fermez puis réouvrez votre session avant de lancer la commande `ssh -X gigondas`.

Dans la fenêtre, saisissez la *passphrase* que vous avez utilisée pour créer la paire de clés publique/privée.

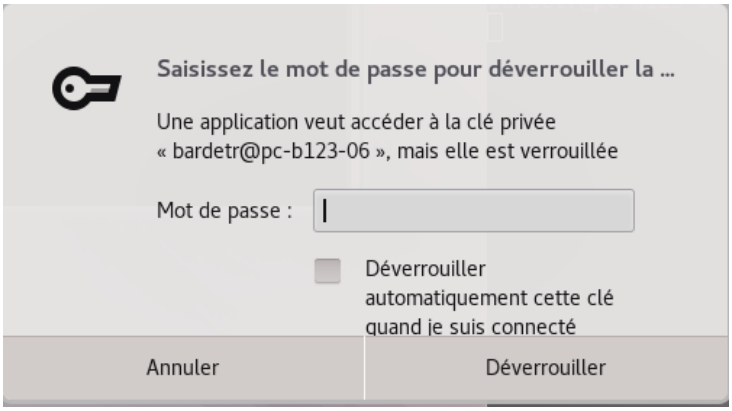


FIGURE 1 – Fenêtre de déverrouillage de la clé privée

Vous devriez maintenant pouvoir ouvrir un terminal sur `gigondas`, sans donner le mot de passe. Essayez par exemple la commande : `ssh -X gigondas xterm`

Pour simplifier les travaux ultérieurs, créez un script `gig` dans votre répertoire `$HOME/bin` qui fonctionne comme suit :

`gig`	sans paramètre ouvre un terminal sur gigondas
`gig ls /etc`	avec des paramètres, suppose que les paramètres sont une commande à exécuter sur `gigondas`.

Testez ce script dans les deux cas donnés ci-dessus.

La commande `ssh-add` permet d'ajouter manuellement une clé à un agent, mais on l'utilise rarement à cette fin puisque l'environnement Gnome offre des méthodes automatiques pour ça. Par contre, on peut obtenir la liste des clés enregistrées avec :

```
ssh-add -l
```

Testez cette commande dans une fenêtre du PC de la salle de TP.

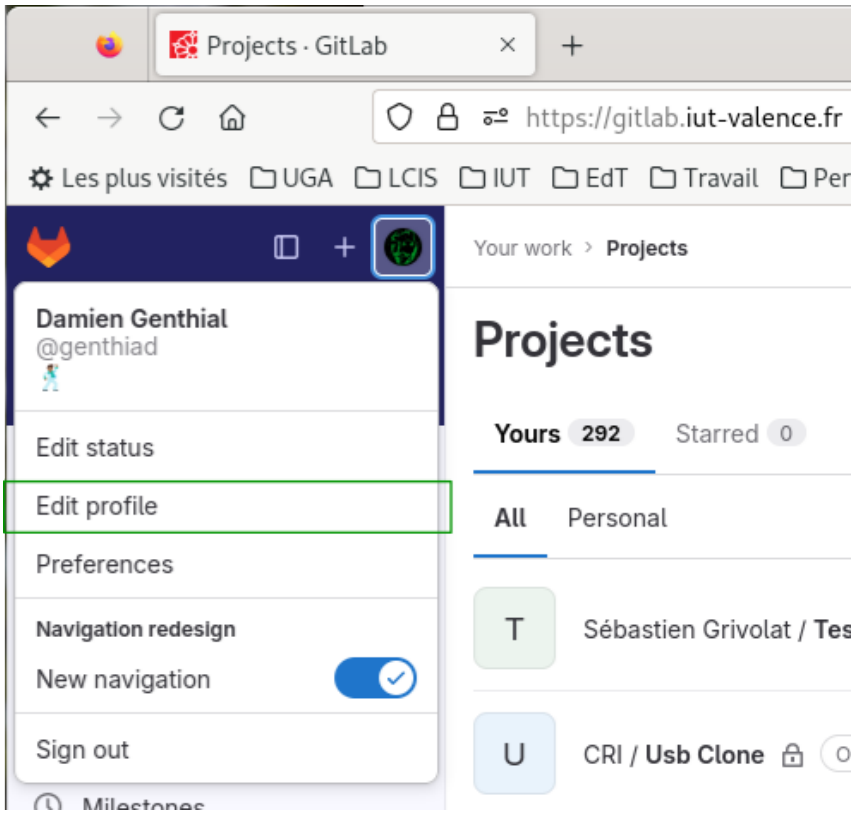
2 Installation de votre clé dans Gitlab

Quand vous utilisez `git` pour tirer ou pousser des modifications dans un projet Gitlab, vous devez donner à chaque fois votre identifiant et votre mot de passe. On peut éviter cet inconvénient en installant votre clé publique dans votre compte Gitlab.

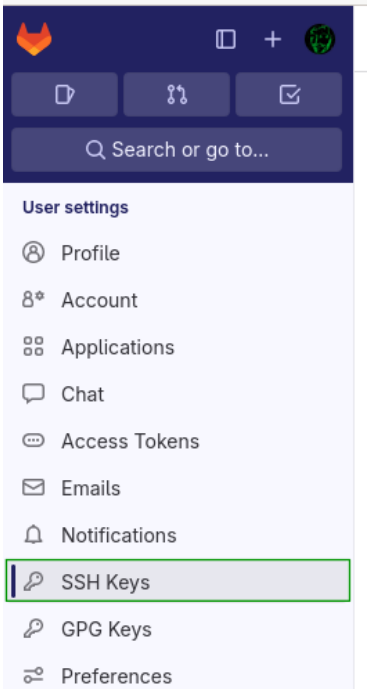
Ouvrez Gitlab : https://gitlab.iut-valence.fr

1. Je vous laisse réfléchir à cette question...

Ouvrez la configuration de votre compte : dans le menu tout en haut à droite, sélectionnez Edit profile.



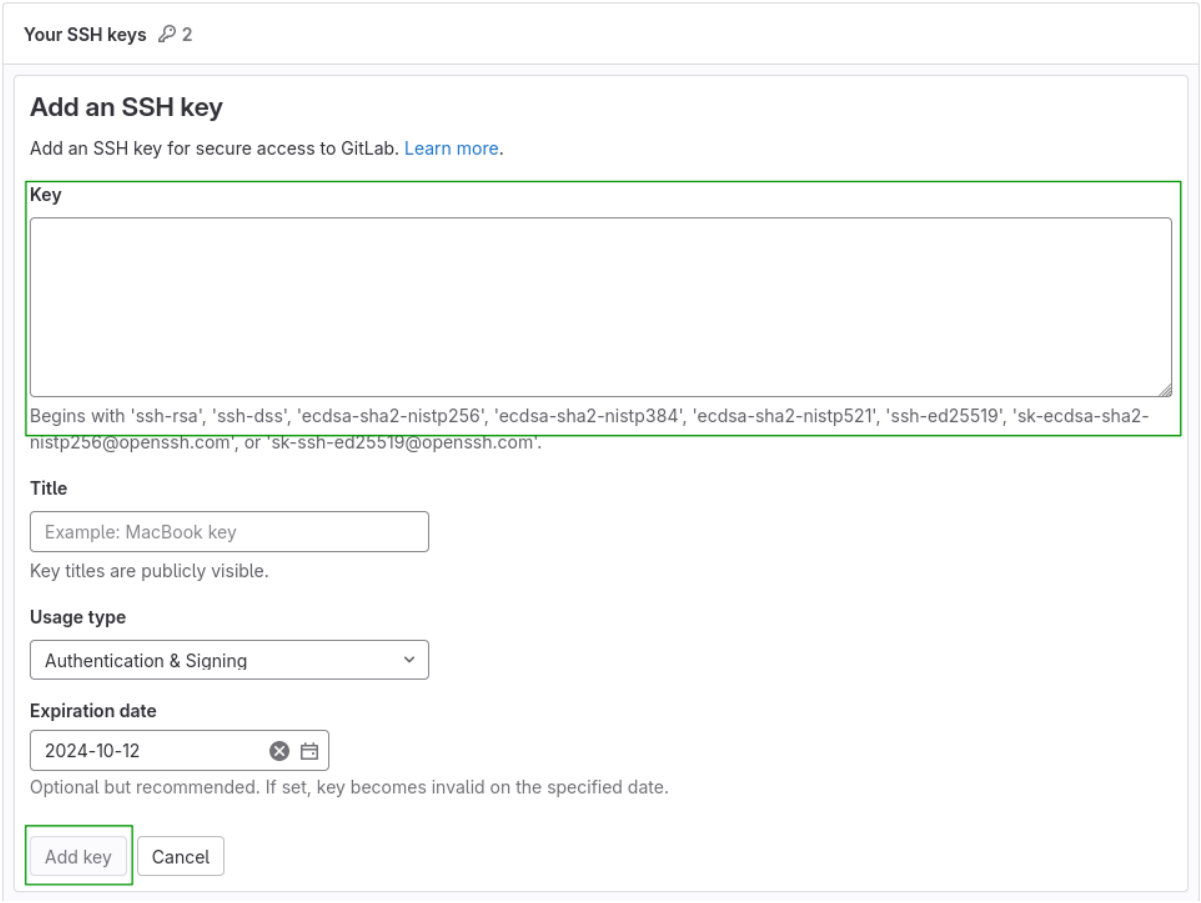
Un menu apparaît à gauche, cliquez sur SSH Keys dans ce menu puis sur la fenêtre de droite cliquez sur Add new key.



Revenez dans votre terminal (`Alt-Tab`) puis ouvrez votre clé publique par exemple avec `Gedit` :
`gedit $HOME/.ssh/id_rsa.pub`

Dans Gedit sélectionnez tout avec `Ctrl-A` puis copiez la clé avec `Ctrl-C`

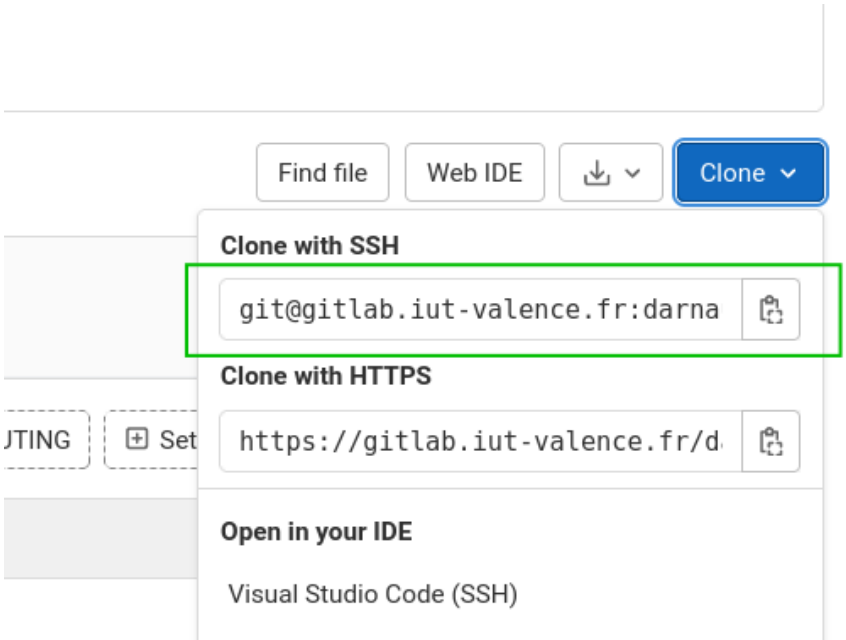
Revenez à Gitlab (toujours `Alt-Tab`) et collez la clé (`Ctrl-V` dans la zone Key) et enregistrez avec Add key (voir figure ci-dessous).



Votre clé est maintenant utilisable.

Il vous reste à modifier la configuration Git de votre projet pour qu'il utilise une connexion par SSH plutôt qu'une connexion par HTTPS. Nous allons faire un essai avec le projet `gitlab-bases` que vous avez créé avec M. Jean dans le cours d'introduction à la gestion de version.

Dans Gitlab ouvrez votre projet `gitlab-bases` et sélectionner l'URL de clonage par SSH puis cliquer sur le bouton pour copier l'URL.



Naviguez dans vos documents jusqu'au dossier contenant le clone du projet, qui s'appelle en principe `gitlab-bases`. Avec votre éditeur favori (sur l'exemple j'utilise `vim` mais vous pouvez utiliser `geany`, `vscode` ou `gedit`), ouvrez le fichier `.git/config` :

```
vim .git/config
```

Sur la ligne contenant `url = ...` remplacez l'URL `https` par l'URL `ssh` que vous venez de copier puis enregistrez le fichier et quittez l'éditeur.

Essayez `git pull` dans ce répertoire. Il ne devrait pas vous demander le mot de passe.

Créer dans le projet un fichier `R1.04.txt` contenant simplement "Bonjour le monde!". Vous pouvez voir l'état de votre projet avec : `git status`. Le fichier `R1.04.txt` doit apparaître comme nouveau fichier.

Ajoutez le fichier à l'index Git avec : `git add R1.04.txt` (ou encore `git add .` qui ajoute tous les nouveaux fichiers du répertoire courant).

Soumettez les modifications avec `git commit -m "Test clé SSH"` (`-m` permet de donner le message de *commit* directement sur la ligne de commande).

Vous pouvez maintenant pousser les modifications sur votre dépôt Gitlab de référence :

```
git push
```

Pour vos prochains projets Gitlab, utilisez plutôt l'URL `ssh` qui permet de manipuler votre projet en ligne de commande sans donner de mot de passe.

3 Utilisation de SSH pour récupérer les fichiers de votre binôme

Il n'est pas recommandé de donner son mot de passe de session à son binôme, mais il est possible de lui ouvrir un accès à `gigondas` par `scp` afin qu'il puisse recopier les fichiers réalisés en séance sur sa propre session.

Commençons par vous ouvrir un accès au compte de votre binôme :

1. Copiez votre clé publique sur le compte de votre binôme :
   `scp id_rsa.pub` *`loginBinôme`*`@gigondas:`
   Il faudra bien sûr que votre binôme saisisse son mot de passe.
2. Connectez-vous au compte de votre binôme par `ssh` :
   `ssh` *`loginBinôme`*`@gigondas`
3. Sur le compte de votre binôme, ajoutez votre clé publique à son fichier `.ssh/authorized_keys` :
   `cat id_rsa.pub >> .ssh/authorized_keys`
   NB : créez si nécessaire le répertoire `.ssh`
4. Déconnectez-vous du compte de votre binôme.
5. Depuis votre compte, copiez le fichier `gig` sur sa session :
   `scp bin/gig` *`loginBinôme`*`@gigondas:bin`
   Si tout s'est bien passé, vous ne devriez pas avoir à donner le mot de passe.

Faites maintenant l'opération dans l'autre sens :

— Fermez votre session et ouvrez la session de votre binôme.

— Créez une paire de clés publique/privée avec `ssh-keygen`.

— Ajoutez la clé publique dans `authorized_keys` ; il suffit de faire :

```
cd .ssh
cat id_rsa.pub >> authorized_keys
```

— Testez cette clé en vous connectant à `gigondas` : `ssh -X gigondas`
Vous devriez voir apparaître la fenêtre de déverrouillage de la clé.

— Installez cette clé dans le fichier `.ssh/authorized_keys` de votre binôme.

— Vérifiez en allant chercher le fichier `gig` chez votre binôme :
`scp` *`loginBinôme`*`@gigondas:bin/gig bin`

ATTENTION : cette technique est moins risquée que celle qui consiste à donner votre mot de passe à votre binôme, car il utilise sa propre *passphrase* pour déverrouiller sa clé, et comme il n'a pas votre mot de passe :

— il ne peut pas le changer ;

— il ne peut pas utiliser vos identifiants pour se connecter à l'UGA ou sur l'intranet de l'IUT, ni pour ouvrir une session Windows.

Cependant, il peut ouvrir une session sur `gigondas` comme s'il était vous-même, la confiance reste donc indispensable.

4 Compte-rendu

Aucun compte-rendu n'est demandé pour ce TP. Il est important que vous ayez bien compris l'utilisation des paires de clés publique/privée donc n'hésitez pas à interpeller l'enseignant à la moindre difficulté.

NB : vous pouvez supprimer l'accès de votre binôme à votre session en supprimant simplement sa clé publique de votre fichier `.ssh/authorized_keys` (avec votre éditeur favori).