Archidep

# Exercice Cours Introduction

## Hello Shell

### Etape 1:

**Étape 1 : Création du répertoire treasure\_hunt**

Nous allons d'abord créer un nouveau répertoire dans votre espace de travail pour débuter la chasse au trésor. Ouvrez Git Bash, puis tapez la commande suivante pour créer un dossier appelé treasure\_hunt :

mkdir treasure\_hunt

Cette commande crée un nouveau répertoire nommé treasure\_hunt dans votre répertoire actuel.

Pour vérifier que le répertoire a bien été créé, vous pouvez utiliser la commande ls :

ls

Vous devriez voir treasure\_hunt apparaître dans la liste.

**Étape 2 : Création des répertoires cave, forest et lake**

Ensuite, nous allons entrer dans le répertoire treasure\_hunt pour créer trois sous-répertoires : cave, forest, et lake. Pour cela, suivez ces étapes :

1. Changez de répertoire vers treasure\_hunt :

cd treasure\_hunt

1. Créez les trois sous-répertoires avec la commande mkdir :

mkdir cave forest lake

1. Pour vérifier que les répertoires ont bien été créés, tapez :

Ls

Vous verrez maintenant les trois répertoires dans treasure\_hunt.

**Étape 3 : Création des fichiers dans les répertoires**

Maintenant, nous allons créer trois fichiers dans les sous-répertoires :

* echo.txt dans cave
* fish.txt dans lake
* tree.txt dans forest

1. **Créer le fichier echo.txt dans cave** :

touch cave/echo.txt

1. **Créer fish.txt dans lake** :

touch lake/fish.txt

1. **Créer tree.txt dans forest** :

touch forest/tree.txt

**Récapitulatif**

* Vous avez créé un répertoire treasure\_hunt avec trois sous-répertoires (cave, forest, et lake).
* Vous avez créé trois fichiers (echo.txt, fish.txt, et tree.txt) dans leurs répertoires respectifs.

Vérifiez que tout est en place en naviguant avec ls dans chaque répertoire !

### Etape 2 :

**Étape 1 : Ajouter les indices dans les fichiers**

Nous allons utiliser la commande echo pour ajouter du texte dans chacun des fichiers que nous avons créés précédemment.

1. **Ajout du texte dans cave/echo.txt** :

echo "Pour découvrir l'indice suivant, explorer où coule l'eau." > cave/echo.txt

Cette commande ajoute le texte directement dans le fichier echo.txt dans le répertoire cave.

1. **Ajout du texte dans lake/fish.txt** :

echo "Aventurez-vous profondément dans les bois pour découvrir le dernier indice." > lake/fish.txt

Cette commande fait la même chose pour le fichier fish.txt dans le répertoire lake.

1. **Ajout du texte dans forest/tree.txt** :

echo "perroquet.live" > forest/tree.txt

Cette commande ajoute le dernier indice dans le fichier tree.txt dans le répertoire forest.

**Étape 2 : Vérification du contenu des fichiers**

Une fois les textes ajoutés, vous pouvez vérifier leur contenu avec la commande cat. Cela affichera le contenu des fichiers dans le terminal.

1. Pour vérifier le fichier cave/echo.txt :

cat cave/echo.txt

1. Pour vérifier le fichier lake/fish.txt :

cat lake/fish.txt

1. Pour vérifier le fichier forest/tree.txt :

cat forest/tree.txt

Chaque commande devrait afficher l'indice que vous avez ajouté dans le fichier correspondant.

**Explication de curl**

La commande **curl** permet de récupérer du contenu à partir d'une URL directement dans le terminal. Par exemple, pour accéder au contenu de perroquet.live, vous pouvez utiliser cette commande :

curl http://perroquet.live

Cela affichera le contenu de la page web directement dans le terminal, sans avoir besoin d'un navigateur. C'est très utile pour interagir avec des serveurs et des API.

**Récapitulatif**

* Vous avez ajouté les indices dans les fichiers avec la commande echo.
* Vous avez utilisé cat pour vérifier que le contenu a bien été ajouté.
* Vous savez maintenant comment utiliser curl pour récupérer des données depuis un serveur web.

Que diriez-vous d'essayer la commande curl pour explorer perroquet.live ?

### Etape 3 :

**Étape 1 : Transition vers le répertoire treasure\_hunt**

Assurez-vous d'être dans le bon répertoire avant de commencer :

cd treasure\_hunt

**Étape 2 : Examiner le contenu de echo.txt dans le répertoire cave**

Le premier indice se trouve dans le fichier echo.txt dans le répertoire cave. Pour lire son contenu, utilisez la commande suivante :

cat cave/echo.txt

Cela affichera le message suivant :

Pour découvrir l'indice suivant, explorer où coule l'eau.

Cet indice vous guide vers le répertoire lake.

**Étape 3 : Naviguer vers le répertoire lake et lire fish.txt**

Suivant l'indice, vous devez vous rendre dans le répertoire lake :

cd lake

Ensuite, pour lire l'indice dans fish.txt, utilisez :

cat fish.txt

Vous verrez le texte suivant :

Aventurez-vous profondément dans les bois pour découvrir le dernier indice.

Cela vous guide vers le répertoire forest.

**Étape 4 : Naviguer vers le répertoire forest et lire tree.txt**

Maintenant, retournez dans le répertoire treasure\_hunt puis entrez dans forest :

cd ..

cd forest

Lisez le fichier tree.txt :

cat tree.txt

Le dernier indice est :

perroquet.live

**Étape 5 : Saisir l'indice final dans le terminal**

Le dernier indice vous indique de visiter le site web perroquet.live. Si vous voulez explorer ce site via la ligne de commande, vous pouvez utiliser curl :

curl http://perroquet.live

Cette commande va vous permettre de voir le contenu de la page dans votre terminal, révélant ainsi le trésor.

**Conclusion**

* Vous avez suivi chaque indice en utilisant les commandes de base comme cat, cd, et curl.
* Vous êtes allé de répertoire en répertoire pour finalement découvrir l'indice final dans le fichier tree.txt.
* Vous avez terminé la chasse au trésor en explorant le site web via la commande curl.

Félicitations, votre chasse au trésor fonctionne parfaitement !

### Etape 4 :

**Étape 1 : Créer le fichier auto\_hunt**

Tout d'abord, nous allons créer un fichier appelé auto\_hunt dans le répertoire treasure\_hunt pour y écrire notre script. Pour cela, utilisez la commande suivante :

touch auto\_hunt

Ensuite, ouvrez ce fichier dans un éditeur de texte comme nano :

nano auto\_hunt

**Étape 2 : Écrire le script pour automatiser la chasse**

Dans l’éditeur, vous allez écrire le script qui guide automatiquement l’utilisateur à travers la chasse au trésor. Le script doit lire chaque indice, introduire une pause avec sleep, puis avancer au prochain fichier.

Voici un exemple de script que vous pouvez écrire :

#!/bin/bash

# Fonction pour lire un fichier et ajouter une pause de 3 secondes

read\_file\_with\_delay() {

cat "$1"

sleep 3

}

# Lire l'indice dans cave/echo.txt

echo "Explorons la caverne..."

read\_file\_with\_delay "cave/echo.txt"

# Lire l'indice dans lake/fish.txt

echo "Allons voir le lac..."

read\_file\_with\_delay "lake/fish.txt"

# Lire l'indice final dans forest/tree.txt

echo "En route vers la forêt..."

read\_file\_with\_delay "forest/tree.txt"

# Exécuter le contenu du fichier tree.txt (dans ce cas, appeler perroquet.live)

echo "Exécution de l'indice final..."

sh forest/tree.txt

**Étape 3 : Sauvegarder et fermer l'éditeur**

Une fois que vous avez terminé d’écrire le script, appuyez sur Ctrl + X, puis Y pour confirmer l’enregistrement, et enfin Entrée pour fermer nano.

**Étape 4 : Donner les droits d’exécution au script**

Avant de pouvoir exécuter le script, vous devez lui donner des droits d’exécution avec la commande suivante :

chmod +x auto\_hunt

**Étape 5 : Exécuter le script**

Maintenant que le script est prêt et exécutable, vous pouvez le lancer avec la commande suivante :

sh auto\_hunt

**Explication du script**

* **Fonction read\_file\_with\_delay** : Cette fonction lit le contenu d'un fichier avec cat, puis ajoute une pause de 3 secondes avec sleep. Cela permet de laisser du temps pour lire chaque indice avant de passer au suivant.
* **Navigation automatique** : Le script suit chaque indice dans l'ordre, affichant les fichiers echo.txt, fish.txt, et tree.txt.
* **Exécution du fichier tree.txt** : Le script exécute le contenu de tree.txt (qui contient l'URL perroquet.live) avec sh forest/tree.txt.

**Récapitulatif**

* Vous avez créé un fichier script auto\_hunt et automatisé le processus de lecture des indices.
* Le script utilise la commande sleep pour introduire des pauses et cat pour lire les fichiers.
* Enfin, vous avez exécuté le contenu du fichier final avec sh.

Lancez le script et voyez la chasse au trésor se dérouler automatiquement !

Etape 5 :

Pour cette étape, nous allons nous concentrer sur la résolution du problème "permission denied" et exécuter correctement le script auto\_hunt. Voici les étapes à suivre :

**Étape 1 : Résoudre l'erreur "Permission Denied"**

Le message d'erreur permission denied: ./auto\_hunt signifie que le script n'a pas les permissions d'exécution. Pour y remédier, vous devez donner au script les droits d'exécution en utilisant la commande chmod.

Exécutez la commande suivante :

chmod +x auto\_hunt

Cette commande permet de rendre le fichier exécutable en ajoutant les droits d'exécution au fichier auto\_hunt.

**Étape 2 : Exécuter le script**

Maintenant que vous avez donné les droits d'exécution, vous pouvez lancer le script. Utilisez la commande suivante :

./auto\_hunt

**Explication**

* **chmod +x auto\_hunt** : Cette commande change les permissions du fichier pour qu'il soit exécutable.
* **./auto\_hunt** : Le ./ indique que le script se trouve dans le répertoire courant, et en l’exécutant directement, vous n’avez plus besoin d'utiliser sh avant le nom du script.

**Résultat attendu**

Après avoir corrigé les permissions et exécuté le script, vous devriez voir les messages de sortie que vous avez programmés dans le script, comme :

Explorons la caverne...

To find the next clue, search where the water flows

Allons voir le lac...

Aventurez-vous profondément dans les bois pour découvrir le dernier indice

En route vers la forêt...

Exécution de l'indice final...

### Etape 6 :

**1. Comprendre le message d'erreur**

Lorsque vous tapez simplement auto\_hunt dans votre terminal et recevez le message command not found: auto\_hunt, cela signifie que le terminal ne sait pas où chercher le script. Cela se produit parce que le répertoire contenant le script (~/treasure\_hunt) n'est pas inclus dans la variable d'environnement PATH, qui liste les dossiers où le terminal cherche les commandes.

**2. Vérification du contenu de votre PATH**

Vous pouvez afficher les répertoires actuellement dans votre PATH en tapant :

echo $PATH

Cela retournera une liste de chemins séparés par des deux-points. Par exemple : /usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin.

**3. Ajouter ~/treasure\_hunt à votre PATH**

Pour que votre terminal trouve et exécute auto\_hunt sans spécifier le chemin complet, vous devez ajouter le répertoire ~/treasure\_hunt à votre PATH. Voici comment le faire temporairement (pour la session en cours) :

export PATH=$PATH:~/treasure\_hunt

Cela ajoute ~/treasure\_hunt à la liste des répertoires dans lesquels le shell cherche les programmes exécutables.

**4. Tester la commande auto\_hunt**

Après avoir ajouté le répertoire au PATH, essayez d'exécuter la commande :

auto\_hunt

Vous devriez maintenant voir le script s'exécuter normalement. Par exemple :

Explorons la caverne...

To find the next clue, search where the water flows

...

**5. Rendre la modification permanente**

Le problème avec la commande export est qu'elle est temporaire et sera réinitialisée lorsque vous fermez la session. Pour rendre l'ajout de ~/treasure\_hunt permanent, vous devez modifier le fichier d'initialisation de votre shell (comme ~/.bashrc ou ~/.bash\_profile pour Bash, ou ~/.zshrc pour Zsh sur macOS).

Ouvrez le fichier avec un éditeur de texte, par exemple avec nano :

nano ~/.bashrc

Ajoutez la ligne suivante à la fin du fichier :

export PATH=$PATH:~/treasure\_hunt

Sauvegardez et quittez en appuyant sur Ctrl + X, puis Y pour confirmer l'enregistrement, et enfin sur Entrée.

**6. Appliquer les modifications sans redémarrer le terminal**

Pour que le terminal prenne en compte ces modifications sans avoir à le redémarrer, vous pouvez "sourcer" le fichier d'initialisation :

source ~/.bashrc

Pour Zsh, ce serait :

source ~/.zshrc

**7. Exécuter le script auto\_hunt depuis n'importe où**

À ce stade, vous pouvez maintenant exécuter le script auto\_hunt de n'importe où dans votre terminal, sans avoir à spécifier le chemin complet. Essayez-le à nouveau :

auto\_hunt

Cela doit maintenant fonctionner peu importe votre emplacement dans le système de fichiers.

**Conclusion**

En suivant ces étapes, vous avez ajouté le répertoire de votre script au PATH, permettant ainsi à votre terminal de l'exécuter directement comme n'importe quelle autre commande. Vous avez appris à temporairement et définitivement modifier la variable PATH pour rendre vos scripts plus facilement exécutables.

## Bonjour SSH

### Etape 1 :

Pour commencer, la commande SSH te permet de te connecter à un serveur distant via un terminal sécurisé. La commande générale est :

ssh <username>@<hostname>

Dans ton cas, les détails sont :

* **Nom d’utilisateur** : tid
* **Mot de passe** : f24b2d5afc
* **Serveur (hostname)** : archidep.ch

Donc la commande à taper dans ton terminal Git Bash sera :

ssh tid@archidep.ch

**Étape 2 : Avertissement initial de connexion SSH**

Quand tu te connectes à un serveur pour la première fois, tu vas recevoir un message d’avertissement du type :

The authenticity of host 'archidep.ch (x.x.x.x)' can't be established.

RSA key fingerprint is SHA256:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.

Are you sure you want to continue connecting (yes/no)?

Cet avertissement te signale que le terminal ne reconnaît pas encore le serveur distant, car c’est la première fois que tu te connectes à ce serveur. **Il est important de vérifier l’empreinte de clé (fingerprint)** avant de répondre "yes".

L'enseignant aurait dû te fournir cette empreinte de clé (en général un long texte alphanumérique). Assure-toi qu’elle correspond à celle affichée par ton terminal pour éviter une attaque de type "man-in-the-middle" (où un attaquant intercepterait la connexion).

Si l'empreinte est correcte, tu peux répondre yes pour accepter la connexion.

**Étape 3 : Entrer le mot de passe**

Après avoir accepté la connexion, le terminal va te demander d’entrer ton mot de passe. Le mot de passe que tu dois utiliser est : f24b2d5afc.

**Important** : Quand tu tapes le mot de passe dans le terminal, les caractères ne vont pas s’afficher (ceci est une mesure de sécurité). Ne t’inquiète pas si rien ne semble apparaître à l’écran.

Appuie sur **Entrée** après avoir collé ou tapé le mot de passe.

**Étape 4 : Connexion réussie**

Si tu as bien suivi les étapes et que le mot de passe est correct, tu devrais voir une **bannière de bienvenue** s’afficher dans le terminal, quelque chose comme :

Welcome to archidep.ch

You are now connected to the server.

Cela indique que tu es bien connecté au serveur distant. L'invite de commande va également changer pour refléter le fait que tu es maintenant sur un autre système (le serveur). Par exemple, l'invite pourrait ressembler à ceci :

tid@archidep:~$

Toutes les commandes que tu exécutes à partir de maintenant seront **exécutées sur le serveur distant**, et non plus sur ta machine locale.

**Résumé des étapes**

1. **Ouvre Git Bash** (ou un terminal de ton choix).
2. Tape la commande suivante pour te connecter :

bash

Copier le code

ssh tid@archidep.ch

1. Vérifie l’empreinte de clé (fingerprint) avant de répondre "yes".
2. Entre ton mot de passe f24b2d5afc lorsque le terminal le demande (rappelle-toi que rien ne s’affichera lorsque tu tapes ou colles le mot de passe).
3. Tu es maintenant connecté au serveur distant.

### Etape 2 :

**1. Se déconnecter du serveur**

Si tu es toujours connecté au serveur distant, déconnecte-toi d’abord en tapant :

exit

Tu peux aussi ouvrir une nouvelle fenêtre de terminal si tu veux garder l’autre session ouverte.

**2. Créer un fichier texte sur ta machine locale**

Pour créer un fichier texte simple sur ta machine locale, utilise la commande suivante :

echo World > hello.txt

Cette commande va créer un fichier texte appelé **hello.txt** avec le mot "World" dedans.

**3. Copier le fichier localement avec la commande cp**

La commande **cp** est utilisée pour copier des fichiers localement. Faisons un test en copiant le fichier **hello.txt** vers un autre fichier appelé **hello2.txt** :

cp hello.txt hello2.txt

Vérifie que le fichier a été bien copié en listant les fichiers dans le répertoire courant :

ls

Tu devrais voir quelque chose comme :

Copier le code

hello.txt hello2.txt

Cela signifie que la copie a bien fonctionné.

**4. Copier un fichier vers le serveur distant avec la commande scp**

La commande **scp** permet de copier des fichiers vers un serveur distant via SSH. Elle utilise la même syntaxe de connexion que la commande **ssh**. Dans ton cas, tu dois remplacer jdoe par ton nom d’utilisateur, soit **tid**.

Voici la commande pour copier le fichier **hello.txt** sur le serveur distant :

scp hello.txt tid@archidep.ch:hello.txt

Cela signifie :

* **hello.txt** : le fichier que tu veux copier (depuis ta machine locale).
* **tid@archidep.ch:hello.txt** : l’emplacement sur le serveur où tu veux copier le fichier. Ici, on copie dans le répertoire personnel de l’utilisateur **tid**, et le fichier aura le même nom **hello.txt**.

Lorsque tu exécutes cette commande, tu devrais voir une sortie du type :

hello.txt 100% 4 0.6KB/s 00:00

Cela signifie que la copie a réussi.

**5. Vérifier que le fichier a été copié sur le serveur**

Pour vérifier que le fichier **hello.txt** a bien été copié sur le serveur, reconnecte-toi au serveur avec la commande **ssh** :

ssh tid@archidep.ch

Une fois connecté, utilise la commande **ls** pour lister les fichiers dans ton répertoire personnel :

ls

Tu devrais voir **hello.txt** dans la liste. Ensuite, pour afficher le contenu du fichier, utilise la commande **cat** :

cat hello.txt

La sortie devrait être :

World

Enfin, tu peux te déconnecter du serveur en tapant à nouveau :

Exit

**6. Copier un fichier depuis le serveur vers ta machine locale**

Maintenant, nous allons faire l’inverse : copier un fichier **du serveur vers ta machine locale**. La commande est similaire, mais dans l'autre sens. Voici comment copier le fichier **hello.txt** depuis le serveur vers un fichier **hello3.txt** sur ta machine locale :

scp tid@archidep.ch:hello.txt hello3.txt

Cela va copier le fichier **hello.txt** depuis le serveur et le renommer **hello3.txt** sur ta machine locale.

Tu peux vérifier que le fichier a bien été copié avec la commande **cat** :

cat hello3.txt

La sortie sera encore une fois :

World

**7. Exemples supplémentaires d’utilisation de scp**

Voici quelques autres exemples utiles de la commande scp :

* Copier un fichier local vers un répertoire spécifique sur le serveur distant :

scp foo.txt tid@archidep.ch:/tmp/foo.txt

Cette commande copie **foo.txt** dans le répertoire **/tmp** du serveur.

* Copier un répertoire entier de manière récursive :

scp -r my\_folder tid@archidep.ch:/tmp/my\_folder

La commande **-r** permet de copier un répertoire et tout son contenu (fichiers et sous-répertoires).

**Résumé des commandes :**

1. **Créer un fichier local** :

echo World > hello.txt

1. **Copier localement** :

cp hello.txt hello2.txt

1. **Copier vers le serveur distant** :

scp hello.txt tid@archidep.ch:hello.txt

1. **Copier depuis le serveur vers ta machine locale** :

scp tid@archidep.ch:hello.txt hello3.txt

### Etape 3 :

**1. Installer un client SFTP**

Tu peux utiliser l'un des clients SFTP suivants pour te connecter au serveur distant :

* **FileZilla** (disponible sur Windows, Mac et Linux)
* **WinSCP** (pour Windows)
* **Cyberduck** (disponible sur Mac et Windows)

Télécharge et installe l'un de ces clients depuis leur site officiel.

**2. Configurer une connexion SFTP**

Une fois que tu as installé ton client SFTP, tu dois configurer une connexion avec les informations que tu as reçues (nom d'utilisateur, mot de passe, etc.). Voici les étapes typiques pour configurer cette connexion :

**Pour FileZilla :**

* Ouvre **FileZilla**.
* Clique sur **Fichier > Gestionnaire de sites**.
* Clique sur **Nouveau site** et donne-lui un nom (par exemple, "archidep.ch").
* Dans le panneau de droite, entre les informations suivantes :
  + **Hôte** : archidep.ch
  + **Port** : 22 (c'est le port par défaut pour SFTP/SSH)
  + **Protocole** : sélectionne **SFTP – SSH File Transfer Protocol**
  + **Mode d'authentification** : **Normal**
  + **Nom d'utilisateur** : ton nom d'utilisateur (par exemple, "tid")
  + **Mot de passe** : ton mot de passe (par exemple, "f24b2d5afc")

Ensuite, clique sur **Connexion**.

**Pour WinSCP :**

* Ouvre **WinSCP**.
* Dans la fenêtre principale, entre ces informations :
  + **Protocole** : **SFTP**
  + **Nom d'hôte** : archidep.ch
  + **Port** : 22
  + **Nom d'utilisateur** : ton nom d'utilisateur (par exemple, "tid")
  + **Mot de passe** : ton mot de passe (par exemple, "f24b2d5afc`)

Puis clique sur **Login**.

**Pour Cyberduck :**

* Ouvre **Cyberduck**.
* Clique sur **Ouvrir une connexion**.
* Sélectionne **SFTP (SSH File Transfer Protocol)** dans la liste déroulante.
* Entre les informations suivantes :
  + **Hôte** : archidep.ch
  + **Nom d'utilisateur** : ton nom d'utilisateur (par exemple, "tid")
  + **Mot de passe** : ton mot de passe (par exemple, "f24b2d5afc")
  + **Port** : 22

Ensuite, clique sur **Connexion**.

**3. Accepter l'empreinte digitale de la clé**

Lors de la première connexion, tu verras peut-être un avertissement disant que l'authenticité de l'hôte ne peut pas être vérifiée (comme tu l'as vu avec SSH). Vérifie l'empreinte digitale et accepte-la si elle correspond à celle fournie par ton enseignant.

**4. Copier un fichier vers le serveur**

Maintenant que tu es connecté :

* Tu peux faire glisser et déposer des fichiers depuis ton ordinateur local vers le serveur distant.
* Par exemple, tu peux créer un fichier localement (hello.txt) et le faire glisser vers le répertoire du serveur dans le client SFTP.

**5. Vérifier que le fichier a bien été copié**

Tu peux te reconnecter au serveur via SSH pour vérifier que le fichier a bien été copié. Utilise la commande ls pour lister les fichiers sur le serveur :

$ ssh tid@archidep.ch

$ ls

Tu devrais voir hello.txt dans la liste des fichiers sur le serveur.

**6. Déconnexion**

Une fois que tu as terminé, déconnecte-toi de ton client SFTP ou ferme l'application.

Voilà, tu as maintenant copié un fichier sur un serveur distant en utilisant SFTP. 🎉

### Etape 4 :

Voici les étapes détaillées pour mettre en place l'authentification par clé publique sur un serveur SSH. Cela permet d'améliorer la sécurité et d'éviter d'avoir à entrer ton mot de passe à chaque connexion SSH.

**Étape 1 : Vérifier si tu as déjà une paire de clés SSH**

Ouvre ton terminal (ou Git Bash sur Windows) et exécute cette commande :

$ ls ~/.ssh

Si tu vois des fichiers comme id\_ed25519 et id\_ed25519.pub (ou bien id\_rsa et id\_rsa.pub), cela signifie que tu as déjà une paire de clés SSH. Tu peux directement passer à **l'étape 3**.

Si tu ne vois pas ces fichiers ou si le répertoire .ssh n'existe pas, cela signifie que tu n'as pas encore de clés. Dans ce cas, passe à l'étape suivante pour en générer une.

**Étape 2 : Générer une paire de clés SSH**

Si tu n'as pas encore de clés SSH, voici comment en générer une nouvelle :

1. **Générer une paire de clés avec ssh-keygen :**

Tape cette commande dans ton terminal pour générer une paire de clés SSH (par défaut, ed25519 est plus sécurisé que rsa) :

$ ssh-keygen -t ed25519 -C "ton\_email@example.com"

Si tu veux générer une clé RSA au lieu d'Ed25519 :

$ ssh-keygen -t rsa -b 4096 -C "ton\_email@example.com"

1. **Répondre aux questions de ssh-keygen :**
   * **Où enregistrer la clé ?** Appuie simplement sur Entrée pour accepter l'emplacement par défaut (~/.ssh/id\_ed25519 ou ~/.ssh/id\_rsa).
   * **Mot de passe pour protéger la clé ?** Si tu veux ajouter un mot de passe pour sécuriser davantage ta clé privée, entre un mot de passe. Si tu préfères ne pas avoir de mot de passe (pour éviter de le saisir à chaque utilisation), appuie simplement sur Entrée.
2. **Ta paire de clés a été créée :** La clé publique sera enregistrée sous le nom id\_ed25519.pub (ou id\_rsa.pub), et la clé privée sera id\_ed25519 (ou id\_rsa).

**Étape 3 : Ajouter la clé publique au serveur distant**

Maintenant que ta paire de clés est prête, il faut ajouter ta clé publique au serveur pour permettre l'authentification par clé publique.

1. **Afficher le contenu de ta clé publique :**

$ cat ~/.ssh/id\_ed25519.pub

Ou pour une clé RSA :

$ cat ~/.ssh/id\_rsa.pub

Le terminal affichera quelque chose de semblable à ceci :

ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGvBZ+e4KdkWosxFNbML54Tme5OPkNsyuPZXfkg6UBoh ton\_email@example.com

1. **Copier ta clé publique sur le serveur SSH :**

Pour copier ta clé publique sur le serveur distant, utilise cette commande (en remplaçant tid par ton nom d'utilisateur et archidep.ch par le nom du serveur) :

$ ssh-copy-id tid@archidep.ch

Cette commande ajoutera ta clé publique au fichier ~/.ssh/authorized\_keys du serveur distant.

1. **Saisir ton mot de passe une dernière fois :** Tu devras entrer ton mot de passe SSH une dernière fois pour authentifier la commande.
2. **Vérification de l'authentification par clé publique :**

Maintenant que ta clé publique est sur le serveur, essaie de te reconnecter au serveur sans utiliser de mot de passe :

$ ssh tid@archidep.ch

Si tout s'est bien passé, tu devrais être connecté sans qu'il te soit demandé de saisir ton mot de passe.

**Étape 4 : (Optionnelle) Ajouter la clé privée à l'agent SSH**

Si tu as défini un mot de passe pour ta clé privée, tu peux utiliser l'agent SSH pour éviter de devoir entrer le mot de passe à chaque fois que tu te connectes. Voici comment ajouter ta clé à l'agent SSH :

1. **Démarrer l'agent SSH :**

$ eval "$(ssh-agent -s)"

1. **Ajouter ta clé privée à l'agent :**

$ ssh-add ~/.ssh/id\_ed25519

Si tu utilises une clé RSA :

$ ssh-add ~/.ssh/id\_rsa

Une fois que ta clé est ajoutée à l'agent, tu n'auras plus besoin d'entrer le mot de passe pour cette session.

**Générer une paire de clés publique/privée**

Pour générer une paire de clés publique/privée sur ta machine locale, utilise la commande ssh-keygen :

1. **Exécuter la commande suivante sur ta machine locale** (pas sur le serveur distant) :

$ ssh-keygen

Cela générera une paire de clés en utilisant l'algorithme par défaut (généralement rsa ou ed25519). Le processus ressemble à ceci :

Generating public/private rsa key pair.

Enter file in which to save the key (/home/jdoe/.ssh/id\_rsa):

Created directory '/home/jdoe/.ssh'.

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /home/jdoe/.ssh/id\_rsa.

Your public key has been saved in /home/jdoe/.ssh/id\_rsa.pub.

The key fingerprint is:

SHA256:MmwL9n4KOUCuLoyvGJ7nWRDXjTSGAXO8AcCNVqmDJH0 jdoe@497820feb22a

The key's randomart image is:

+---[RSA 2048]----+

|.o===oo+ |

|.=.oE++ + |

|= oo .oo . |

|.= oo |

| +.o = S |

| . o.= + |

|= +.o |

|\*o..o+ . |

|+\*+o oo |

+----[SHA256]-----+

1. **Choisir un emplacement pour sauvegarder la clé** :

Quand il te demande "Enter file in which to save the key", appuie simplement sur **Entrée** pour accepter l'emplacement par défaut (généralement ~/.ssh/id\_rsa ou ~/.ssh/id\_ed25519).

1. **Saisir une phrase de passe (optionnelle)** :

Si tu veux protéger ta clé avec une phrase de passe, entre-la lorsque le système te la demande. Sinon, appuie simplement sur **Entrée** pour ignorer cette option.

1. **Vérifier que la paire de clés a bien été générée** :

Après avoir généré ta clé, tu peux vérifier qu'elle a bien été créée en listant les fichiers dans ton répertoire .ssh :

$ ls ~/.ssh

Tu devrais voir deux fichiers : la clé privée (id\_rsa ou id\_ed25519) et la clé publique (id\_rsa.pub ou id\_ed25519.pub).

**Copier ta clé publique sur le serveur SSH**

Maintenant que tu as généré ta paire de clés, tu dois copier la **clé publique** sur le serveur SSH.

1. **Utilise ssh-copy-id pour copier ta clé publique sur le serveur** :

Utilise la commande suivante (en remplaçant jdoe par ton nom d'utilisateur sur le serveur et archidep.ch par le nom du serveur) :

$ ssh-copy-id jdoe@archidep.ch

Cette commande copie automatiquement ta clé publique dans le fichier ~/.ssh/authorized\_keys du serveur distant.

1. **Saisir ton mot de passe SSH une dernière fois** :

Lorsque tu exécutes ssh-copy-id, il te demandera ton mot de passe pour se connecter au serveur et copier la clé.

1. **Se connecter sans mot de passe** :

Une fois que la clé est copiée, tu peux te reconnecter au serveur en utilisant simplement la commande SSH habituelle, et tu ne devrais plus avoir besoin de saisir ton mot de passe :

$ ssh jdoe@archidep.ch

Si tout est bien configuré, tu seras connecté sans mot de passe. Le serveur utilise désormais ta clé publique pour authentifier la connexion, et ton client SSH utilise la clé privée correspondante.

**Vérifier le fichier authorized\_keys sur le serveur**

Si tu veux vérifier que ta clé publique a bien été copiée sur le serveur :

1. **Se connecter au serveur** :

Si tu n'es pas déjà connecté, utilise ssh pour te connecter au serveur :

$ ssh jdoe@archidep.ch

1. **Lister le contenu du répertoire .ssh** :

Tape cette commande pour vérifier que le fichier authorized\_keys est présent :

$ ls ~/.ssh

Tu devrais voir le fichier authorized\_keys.

1. **Afficher le contenu de authorized\_keys** :

Tu peux également afficher le contenu de ce fichier pour t'assurer que ta clé publique y a été ajoutée :

$ cat ~/.ssh/authorized\_keys

Tu devrais voir ta clé publique, qui ressemblera à quelque chose comme ceci :

ssh-rsa AAAAB3NzaC1yc2EAA... example

**Configurer manuellement le fichier authorized\_keys (optionnel)**

Si, pour une raison quelconque, ssh-copy-id ne fonctionne pas ou si tu préfères le faire manuellement, tu peux aussi créer le fichier authorized\_keys et y ajouter ta clé publique manuellement.

1. **Créer le répertoire .ssh sur le serveur** (si nécessaire) :

Si le répertoire .ssh n'existe pas, tu peux le créer manuellement :

$ mkdir -p ~/.ssh && chmod 700 ~/.ssh

1. **Ajouter ta clé publique à authorized\_keys** :

Utilise la commande suivante pour créer (ou ouvrir) le fichier authorized\_keys et y ajouter ta clé publique :

$ touch ~/.ssh/authorized\_keys && chmod 600 ~/.ssh/authorized\_keys

Ensuite, ajoute ta clé publique au fichier authorized\_keys avec cette commande :

$ echo "ssh-rsa AAAAB3NzaC1yc2EAA..." >> ~/.ssh/authorized\_keys

Assure-toi que les permissions du fichier et du répertoire sont correctes, sinon le serveur SSH refusera d'utiliser la clé pour des raisons de sécurité.