Tutoriels DigitalOcean.

Source : https://www.digitalocean.com/community/tutorials?q=Debian+10

# Configuration initiale du serveur avec Debian 10.

## Introduction

Lorsque vous créez un nouveau serveur Debian 10, il y a quelques étapes de configuration que vous devez suivre dès le départ dans le cadre de l'installation de base. Cela augmentera la sécurité et la facilité d'utilisation de votre serveur et vous donnera une base solide pour les actions ultérieures.

Dans ce tutoriel, nous apprendrons à nous connecter à notre serveur en tant qu'utilisateur **root**, à créer un nouvel utilisateur avec des privilèges d'administrateur et à mettre en place un pare-feu de base.

## Étape 1 - Connexion en tant que root

Pour vous connecter à votre serveur, vous devez connaître **l'adresse IP publique de votre serveur**. Vous aurez également besoin du mot de passe ou, si vous avez installé une clé SSH pour l'authentification, de la clé privée du compte de l'utilisateur **root**. Si vous n'avez pas encore ouvert de session sur votre serveur, vous pouvez suivre notre guide sur la façon de vous connecter à votre Droplet avec SSH, qui couvre ce processus en détail.

Si vous n'êtes pas encore connecté à votre serveur, allez-y et connectez-vous en tant qu'utilisateur **root** en utilisant la commande suivante (remplacez la partie surlignée de la commande par l'adresse IP publique de votre serveur) :

$ ssh root@your\_server\_ip

Acceptez l'avertissement concernant l'authenticité de l'hôte s'il apparaît. Si vous utilisez l'authentification par mot de passe, fournissez votre mot de passe root pour vous connecter. Si vous utilisez une clé SSH protégée par une phrase de passe, il se peut que vous soyez invité à entrer la phrase de passe la première fois que vous utilisez la clé à chaque session. Si c'est la première fois que vous vous connectez au serveur avec un mot de passe, vous pouvez également être invité à changer le mot de passe **root**.

## À propos de root

L'utilisateur **root** est l'utilisateur administratif dans un environnement Linux qui a des privilèges très larges. En raison des privilèges accrus du compte **root**, il est déconseillé de l'utiliser régulièrement. Cela est dû au fait qu'une partie du pouvoir inhérent au compte **root** est la capacité à effectuer des modifications très destructrices, même par accident.

L'étape suivante consiste à créer un autre compte utilisateur avec un champ d'influence réduit pour le travail quotidien. Plus tard, nous vous expliquerons comment obtenir des privilèges accrus pour les moments où vous en avez besoin.

### changement du mot de passe root  
passwd  
  
apt update  
apt upgrade  
  
### Paquets complémentaires  
  
apt install ntp ssh openssh-server vim-nox zip ntp zsh arj unzip screenfetch curl git bzip2 lzma xpdf binutils nomarch lzop cabextract  
  
### Config Zsh et Vim  
  
wget https://formation-debian.viarezo.fr/fichiers-config.tar.gz  
tar xvzf fichiers-config.tar.gz  
cd fichiers-config  
cp vimrc /etc/vim/  
cp zshrc zshenv zlogin zlogout /etc/zsh/  
cp dir\_colors /etc/  
  
vim /etc/adduser.conf  
chsh <= /bin/zsh  
  
Supprimer les fichiers qui figurent dans le répertoire /etc/skel  
  
Corriger le fichier /etc/zsh/zshenv  
  
déconnection et reconnection  
================================================================================  
sudo apt install gnupg2 ca-certificates lsb-release  
----------  
echo "deb http://nginx.org/packages/debian `lsb\_release -cs` nginx" | sudo tee /etc/apt/sources.list.d/nginx.list  
----------  
curl -fsSL https://nginx.org/keys/nginx\_signing.key | sudo apt-key add -  
----------  
Verify that you now have the proper key:  
  
sudo apt-key fingerprint ABF5BD827BD9BF62  
  
The output should contain the full fingerprint 573B FD6B 3D8F BC64 1079 A6AB ABF5 BD82 7BD9 BF62 as follows:  
  
 pub rsa2048 2011-08-19 [SC] [expires: 2024-06-14]  
 573B FD6B 3D8F BC64 1079 A6AB ABF5 BD82 7BD9 BF62  
 uid [ unknown] nginx signing key <signing-key@nginx.com>  
----------  
sudo apt update  
sudo apt install nginx  
  
  
================================================================================

## Étape 2 - Création d'un nouvel utilisateur

Une fois que vous êtes connecté en tant que **root**, nous sommes prêts à ajouter le nouveau compte d'utilisateur que nous utiliserons pour nous connecter à partir de maintenant.

Cet exemple crée un nouvel utilisateur appelé **serveradm**, mais vous devez le remplacer par un nom d'utilisateur que vous aimez :

# adduser serveradm

Quelques questions vous seront posées, à commencer par le mot de passe du compte.

Entrez un mot de passe fort et, éventuellement, complétez les informations supplémentaires que vous souhaitez. Cela n'est pas obligatoire et vous pouvez simplement appuyer sur ENTER dans n'importe quel champ que vous souhaitez ignorer.

Ensuite, nous allons configurer ce nouvel utilisateur avec des privilèges d'administrateur.

## Étape 3 - Octroi de privilèges administratifs

Nous avons maintenant créé un nouveau compte d'utilisateur avec des privilèges de compte réguliers. Cependant, nous pouvons parfois avoir besoin d'effectuer des tâches administratives avec ce compte.

Pour éviter d'avoir à nous déconnecter de notre utilisateur normal et à nous reconnecter en tant que compte **root**, nous pouvons configurer ce que l'on appelle les privilèges de *super-utilisateur* ou de **root** pour notre compte normal. Cela permettra à notre utilisateur normal d'exécuter des commandes avec des privilèges administratifs en mettant le mot sudo avant la commande.

Pour ajouter ces privilèges à notre nouvel utilisateur, nous devons ajouter le nouvel utilisateur au groupe **sudo**. Par défaut, sous Debian 10, les utilisateurs qui appartiennent au groupe **sudo** sont autorisés à utiliser la commande sudo.

En tant que **root**, exécutez cette commande pour ajouter votre nouvel utilisateur au groupe **sudo** (remplacez le mot en surbrillance par votre nouvel utilisateur) :

# usermod -aG sudo serveradm  
(ou)  
# adduser serveradm sudo

Maintenant, lorsque vous êtes connecté en tant qu'utilisateur habituel, vous pouvez taper sudo avant les commandes pour exécuter la commande avec les privilèges de super-utilisateur.

## Étape 4 - Mise en place d'un pare-feu de base

Les serveurs Debian peuvent utiliser des pares-feux pour s'assurer que seules certaines connexions à des services spécifiques sont autorisées. Dans ce guide, nous allons installer et utiliser le pare-feu UFW pour aider à définir les règles du pare-feu et à gérer les exceptions.

Nous pouvons utiliser le gestionnaire de paquets apt pour installer UFW. Mettez à jour l'index local pour récupérer les dernières informations sur les paquets disponibles, puis installez le logiciel de pare-feu UFW en tapant :

# apt update  
# apt install ufw

Les profils de pare-feu permettent à l'UFW de gérer des ensembles de règles de pare-feu nommés pour les applications installées. Les profils de certains logiciels courants sont fournis avec UFW par défaut et les logiciels peuvent enregistrer des profils supplémentaires avec UFW pendant le processus d'installation. OpenSSH, le service qui nous permet de nous connecter à notre serveur maintenant, possède un profil de pare-feu que nous pouvons utiliser.

Vous pouvez lister tous les profils d'application disponibles en tapant :

# ufw app list

Output  
Available applications:  
 . . .  
 OpenSSH  
 . . .

Nous devons nous assurer que le pare-feu autorise les connexions SSH afin que nous puissions nous reconnecter la prochaine fois. Nous pouvons autoriser ces connexions en tapant :

# ufw allow OpenSSH

Ensuite, nous pouvons activer le pare-feu en tapant :

# ufw enable

Tapez y et appuyez sur ENTER pour continuer. Vous pouvez voir que les connexions SSH sont toujours autorisées en tapant :

# ufw status

Output  
Status: active  
  
To Action From  
-- ------ ----  
OpenSSH ALLOW Anywhere  
OpenSSH (v6) ALLOW Anywhere (v6)

Comme le **pare-feu bloque actuellement toutes les connexions, à l'exception de la connexion SSH**, si vous installez et configurez des services supplémentaires, vous devrez ajuster les paramètres du pare-feu pour permettre un trafic d'entrée acceptable. Vous pouvez apprendre quelques opérations UFW courantes dans notre guide UFW essentials.

## Étape 5 - Activation de l'accès externe pour votre utilisateur habituel

Maintenant que nous avons un utilisateur régulier pour l'usage quotidien, nous devons nous assurer que nous pouvons SSH dans le compte directement.

Note : Jusqu'à ce que nous ayons vérifié que vous pouvez vous connecter et utiliser sudo avec votre nouvel utilisateur, nous vous recommandons de rester connecté en tant que **root**. De cette façon, si vous avez des problèmes, vous pouvez les résoudre et effectuer les changements nécessaires en tant que root.

Le processus de configuration de l'accès SSH pour votre nouvel utilisateur dépend de l'utilisation par le compte **root** de votre serveur d'un mot de passe ou de clés SSH pour l'authentification.

### Si le compte racine utilise l'authentification par mot de passe

Si vous vous êtes connecté à votre compte root à l'aide d'un mot de passe, alors l'authentification par mot de passe est activée pour SSH. Vous pouvez accéder à votre nouveau compte utilisateur en ouvrant une nouvelle session de terminal et en utilisant SSH avec votre nouveau nom d'utilisateur :

$ ssh serveradm@your\_server\_ip

Après avoir saisi votre mot de passe d'utilisateur habituel, vous serez connecté. N'oubliez pas que si vous devez exécuter une commande avec des privilèges administratifs, tapez sudo avant comme ceci :

$ sudo command\_to\_run

Le mot de passe d'utilisateur habituel vous sera demandé lors de la première utilisation de sudo à chaque session (et périodiquement par la suite).

Pour renforcer la sécurité de votre serveur, **nous vous recommandons vivement de configurer des clés SSH au lieu d'utiliser l'authentification par mot de passe**. Suivez notre guide sur la configuration des clés SSH sous Debian 10 pour savoir comment configurer l'authentification par clé.

### Si le compte racine utilise l'authentification par clé SSH

Si vous vous êtes connecté à votre compte **root** en *utilisant des clés SSH*, l'authentification par mot de passe est alors désactivée pour SSH. Vous devrez ajouter une copie de votre clé publique locale au fichier clé ~/.ssh/ authorized\_keys du nouvel utilisateur pour vous connecter.

Comme votre clé publique se trouve déjà dans le fichier ~/.ssh/ authorized\_keys du compte **root** sur le serveur, nous pouvons copier ce fichier et la structure des répertoires sur notre nouveau compte utilisateur dans notre session existante avec la commande cp. Ensuite, nous pouvons ajuster la propriété des fichiers à l'aide de la commande chown.

Veillez à modifier les parties surlignées de la commande ci-dessous pour qu'elles correspondent au nom de votre utilisateur habituel :

# cp -r ~/.ssh /home/serveradm  
# chown -R serveradm:serveradm /home/serveradm/.ssh

La commande cp -r copie l'intégralité du répertoire dans le répertoire personnel du nouvel utilisateur, et la commande chown -R change le propriétaire de ce répertoire (et de tout ce qui s'y trouve) pour le username:groupname spécifié (Debian crée un groupe avec le même nom que votre nom d'utilisateur par défaut).

Maintenant, ouvrez une nouvelle session de terminal et connectez-vous via SSH avec votre nouveau nom d'utilisateur :

$ ssh serveradm@your\_server\_ip

Après avoir saisi votre mot de passe d'utilisateur habituel, vous serez connecté. N'oubliez pas que si vous devez exécuter une commande avec des privilèges administratifs, tapez sudo avant comme ceci :

$ sudo command\_to\_run

Le mot de passe d'utilisateur habituel vous sera demandé lors de la première utilisation de sudo à chaque session (et périodiquement par la suite).

# Comment installer des clés SSH sur Debian 10.

## Introduction

SSH, ou secure shell, est un protocole crypté utilisé pour administrer et communiquer avec des serveurs. Lorsque vous travaillez avec un serveur Debian, il y a de fortes chances que vous passiez la plupart de votre temps dans une session terminale connectée à votre serveur par SSH.

Dans ce guide, nous allons nous concentrer sur la configuration des clés SSH pour une installation Debian 10 de type "vanilla". Les clés SSH fournissent un moyen facile et sûr de se connecter à votre serveur et sont recommandées pour tous les utilisateurs.

## Étape 1 - Création de la paire de clés RSA

La première étape consiste à créer une paire de clés sur la machine cliente (généralement votre ordinateur) :

$ ssh-keygen

Par défaut, ssh-keygen créera une paire de clés RSA de 2048 bits, ce qui est suffisamment sûr pour la plupart des cas d'utilisation (vous pouvez éventuellement passer le drapeau -b 4096 pour créer une clé de 4096 bits plus grande).

Après avoir entré la commande, vous devriez voir la sortie suivante :

Output  
/home/your\_home/.ssh/id\_rsa already exists.  
Overwrite (y/n)?

Appuyez sur la touche Entrée pour enregistrer la paire de clés dans le sous-répertoire .ssh/ de votre répertoire d'origine, ou indiquez un autre chemin d'accès.

Si vous avez déjà généré une paire de clés SSH, vous pouvez voir l'invite suivante :

Output  
Generating public/private rsa key pair.  
Enter file in which to save the key (/your\_home/.ssh/id\_rsa):

Avertissement : Si vous choisissez d'écraser la clé sur le disque, vous ne pourrez plus vous authentifier à l'aide de la clé précédente. Soyez très prudent lorsque vous choisissez oui, car il s'agit d'un processus destructeur qui ne peut être inversé.

Vous devriez alors voir l'invite suivante :

Output  
Enter passphrase (empty for no passphrase):

Ici, vous pouvez éventuellement entrer un mot de passe sécurisée, ce qui est fortement recommandé. Un mot de passe ajoute une couche de sécurité supplémentaire pour empêcher les utilisateurs non autorisés de se connecter.

Vous devriez alors voir le résultat suivant :

Output  
Your identification has been saved in /your\_home/.ssh/id\_rsa.  
Your public key has been saved in /your\_home/.ssh/id\_rsa.pub.  
The key fingerprint is:  
a9:49:2e:2a:5e:33:3e:a9:de:4e:77:11:58:b6:90:26 username@remote\_host  
The key's randomart image is:  
+--[ RSA 2048]----+  
| ..o |  
| E o= . |  
| o. o |  
| .. |  
| ..S |  
| o o. |  
| =o.+. |  
|. =++.. |  
|o=++. |  
+-----------------+

Vous disposez désormais d'une clé publique et privée que vous pouvez utiliser pour vous authentifier. L'étape suivante consiste à placer la clé publique sur votre serveur afin que vous puissiez utiliser l'authentification par clé SSH pour vous connecter.

## Étape 2 - Copier la clé publique sur le serveur Debian

La façon la plus rapide de copier votre clé publique sur l'hôte Debian est d'utiliser un utilitaire appelé ssh-copy-id. En raison de sa simplicité, cette méthode est fortement recommandée si elle est disponible. Si vous ne disposez pas de ssh-copy-id sur votre machine cliente, vous pouvez utiliser l'une des deux méthodes alternatives fournies dans cette section (copie via SSH avec mot de passe, ou copie manuelle de la clé).

### Copie de la clé publique à l'aide de ssh-copy-id

L'outil ssh-copy-id est inclus par défaut dans de nombreux systèmes d'exploitation, il est donc possible qu'il soit disponible sur votre système local. Pour que cette méthode fonctionne, vous devez déjà disposer d'un accès SSH à votre serveur, basé sur un mot de passe.

Pour utiliser l'utilitaire, il vous suffit de spécifier l'hôte distant auquel vous souhaitez vous connecter et le compte utilisateur auquel vous avez accès par SSH avec un mot de passe. Il s'agit du compte sur lequel votre clé publique SSH sera copiée.

La syntaxe est :

$ ssh-copy-id username@remote\_host

Vous pouvez voir le message suivant :

Output  
The authenticity of host '203.0.113.1 (203.0.113.1)' can't be established.  
ECDSA key fingerprint is fd:fd:d4:f9:77:fe:73:84:e1:55:00:ad:d6:6d:22:fe.  
Are you sure you want to continue connecting (yes/no)? yes

Cela signifie que votre ordinateur local ne reconnaît pas l'hôte distant. Cela se produira la première fois que vous vous connecterez à un nouvel hôte. Tapez "yes" et appuyez sur ENTER pour continuer.

Ensuite, l'utilitaire recherchera sur votre compte local la clé id\_rsa.pub que nous avons créée précédemment. Lorsqu'il trouve la clé, il vous demandera le mot de passe du compte de l'utilisateur distant :

Output  
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed  
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys  
username@203.0.113.1's password:

Saisissez le mot de passe (votre saisie ne sera pas affichée pour des raisons de sécurité) et appuyez sur ENTER. L'utilitaire se connectera au compte sur l'hôte distant en utilisant le mot de passe que vous avez fourni. Il copiera ensuite le contenu de votre clé ~/.ssh/id\_rsa.pub dans un fichier situé dans le répertoire ~/.ssh du compte distant appelé authorized\_keys.

Vous devriez voir la sortie suivante :

Output  
Number of key(s) added: 1  
  
Now try logging into the machine, with: "ssh 'username@203.0.113.1'"  
and check to make sure that only the key(s) you wanted were added.

À ce stade, votre clé id\_rsa.pub a été téléchargée sur le compte distant. Vous pouvez passer à l'étape 3.

### Copie de la clé publique en utilisant SSH

Si vous ne disposez pas de ssh-copy-id, mais que vous avez un accès SSH par mot de passe à un compte sur votre serveur, vous pouvez télécharger vos clés en utilisant une méthode SSH classique.

Nous pouvons le faire en utilisant la commande cat pour lire le contenu de la clé publique SSH sur notre ordinateur local et en l'acheminant par une connexion SSH vers le serveur distant.

De l'autre côté, nous pouvons nous assurer que le répertoire ~/.ssh existe et qu'il a les bonnes permissions sous le compte que nous utilisons.

Nous pouvons alors sortir le contenu que nous avons transféré dans un fichier appelé authorized\_keys dans ce répertoire. Nous utiliserons le symbole de redirection >> pour ajouter le contenu au lieu de l'écraser. Cela nous permettra d'ajouter des clés sans détruire les clés précédemment ajoutées.

La commande complète ressemble à ceci :

$ cat ~/.ssh/id\_rsa.pub | ssh username@remote\_host "mkdir -p ~/.ssh && touch ~/.ssh/authorized\_keys && chmod -R go= ~/.ssh && cat >> ~/.ssh/authorized\_keys"

Vous pouvez voir le message suivant :

Output  
The authenticity of host '203.0.113.1 (203.0.113.1)' can't be established.  
ECDSA key fingerprint is fd:fd:d4:f9:77:fe:73:84:e1:55:00:ad:d6:6d:22:fe.  
Are you sure you want to continue connecting (yes/no)? yes

Cela signifie que votre ordinateur local ne reconnaît pas l'hôte distant. Cela se produira la première fois que vous vous connecterez à un nouvel hôte. Tapez "yes" et appuyez sur ENTER pour continuer.

Ensuite, vous devriez être invité à saisir le mot de passe du compte d'utilisateur distant :

Output  
username@203.0.113.1's password:

Après avoir saisi votre mot de passe, le contenu de votre clé id\_rsa.pub sera copié à la fin du fichier authorized\_keys du compte de l'utilisateur distant. Passez à l'étape 3 si vous avez réussi.

### Copie manuelle de la clé publique

Si vous ne disposez pas d'un accès SSH par mot de passe à votre serveur, vous devrez effectuer le processus ci-dessus manuellement.

Nous ajouterons manuellement le contenu de votre fichier id\_rsa.pub au fichier ~/.ssh/authorized\_keys sur votre machine distante.

Pour afficher le contenu de votre clé id\_rsa.pub, tapez celle-ci dans votre ordinateur local :

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

Vous verrez le contenu de la clé, qui devrait ressembler à ceci :

Output  
ssh-rsa  demo@test

Accédez à votre hôte distant en utilisant la méthode dont vous disposez.

Une fois que vous avez accès à votre compte sur le serveur distant, vous devez vous assurer que le répertoire ~/.ssh existe. Cette commande créera le répertoire si nécessaire, ou ne fera rien s'il existe déjà :

$ mkdir -p ~/.ssh

Maintenant, vous pouvez créer ou modifier le fichier authorized\_keys dans ce répertoire. Vous pouvez ajouter le contenu de votre fichier id\_rsa.pub à la fin du fichier authorized\_keys, en le créant si nécessaire, à l'aide de cette commande :

$ echo public\_key\_string >> ~/.ssh/authorized\_keys

Dans la commande ci-dessus, remplacez la chaîne de caractères public\_key\_string par la sortie de la commande cat ~/.ssh/id\_rsa.pub que vous avez exécutée sur votre système local. Elle doit commencer par ssh-rsa AAAA....

Enfin, nous nous assurerons que le répertoire ~/.ssh et le fichier authorized\_keys ont les permissions appropriées :

$ chmod -R go= ~/.ssh

Cela supprime récursivement toutes les permissions "group" et "other" pour le répertoire ~/.ssh/.

Si vous utilisez le compte root pour configurer les clés d'un compte utilisateur, il est également important que le répertoire ~/.ssh appartienne à l'utilisateur et non à root :

$ chown -R serveradm:serveradm ~/.ssh

Dans ce tutoriel, notre utilisateur s'appelle serveradm, mais vous devez remplacer le nom d'utilisateur approprié par la commande ci-dessus.

Nous pouvons maintenant tenter une authentification sans mot de passe avec notre serveur Debian.

## Étape 3 - Authentification au serveur Debian à l'aide de clés SSH

Si vous avez effectué avec succès l'une des procédures ci-dessus, vous devriez pouvoir vous connecter à l'hôte distant sans le mot de passe du compte distant.

La procédure de base est la même :

$ ssh username@remote\_host

Si c'est la première fois que vous vous connectez à cet hôte (si vous avez utilisé la dernière méthode ci-dessus), vous pouvez voir quelque chose comme ceci :

Output  
The authenticity of host '203.0.113.1 (203.0.113.1)' can't be established.  
ECDSA key fingerprint is fd:fd:d4:f9:77:fe:73:84:e1:55:00:ad:d6:6d:22:fe.  
Are you sure you want to continue connecting (yes/no)? yes

Cela signifie que votre ordinateur local ne reconnaît pas l'hôte distant. Cela se produira la première fois que vous vous connecterez à un nouvel hôte. Tapez "yes" puis appuyez sur ENTER pour continuer.

Si vous n'avez pas fourni de mot de passe pour votre clé privée, vous serez immédiatement connecté. Si vous avez fourni un mot de passe pour la clé privée lors de la création de la clé, vous serez invité à la saisir maintenant (notez que vos frappes ne s'afficheront pas dans la session du terminal pour des raisons de sécurité). Après l'authentification, une nouvelle session devrait s'ouvrir pour vous avec le compte configuré sur le serveur Debian.

Si l'authentification par clé a réussi, continuez à apprendre comment sécuriser davantage votre système en désactivant l'authentification par mot de passe.

## Étape 4 - Désactiver l'authentification par mot de passe sur votre serveur

Si vous avez pu vous connecter à votre compte en utilisant SSH sans mot de passe, vous avez configuré avec succès l'authentification basée sur la clé SSH pour votre compte. Cependant, votre mécanisme d'authentification par mot de passe est toujours actif, ce qui signifie que votre serveur est toujours exposé aux attaques par force brute.

Avant d'effectuer les étapes de cette section, assurez-vous que vous avez configuré l'authentification par clé SSH pour le compte racine sur ce serveur, ou de préférence, que vous avez configuré l'authentification par clé SSH pour un compte non-root sur ce serveur avec des privilèges sudo. Cette étape verrouillera les connexions basées sur un mot de passe, il est donc crucial de s'assurer que vous pourrez toujours obtenir un accès administratif.

Une fois que vous avez confirmé que votre compte distant a des privilèges administratifs, connectez-vous à votre serveur distant avec des clés SSH, soit en tant que root, soit avec un compte ayant des privilèges sudo. Ensuite, ouvrez le fichier de configuration du démon SSH :

$ sudo nano /etc/ssh/sshd\_config

Dans le fichier, recherchez une directive appelée PasswordAuthentication. Celle-ci peut être commentée. Décommentez la ligne et mettez la valeur à "no". Cela désactivera votre capacité à vous connecter via SSH en utilisant les mots de passe des comptes :

Output  
...  
PasswordAuthentication no  
...

Enregistrez et fermez le fichier lorsque vous avez terminé en appuyant sur CTRL + X, puis Y pour confirmer l'enregistrement du fichier, et enfin ENTER pour quitter nano. Pour mettre en œuvre ces changements, nous devons redémarrer le service sshd :

$ sudo service ssh restart

Par mesure de précaution, ouvrez une nouvelle fenêtre de terminal et vérifiez que le service SSH fonctionne correctement avant de fermer cette session :

$ ssh username@remote\_host

Une fois que vous avez vérifié votre service SSH, vous pouvez fermer en toute sécurité toutes les sessions de serveur en cours.

Le démon SSH sur votre serveur Debian ne répond plus qu'aux clés SSH. L'authentification par mot de passe a été désactivée avec succès.

## Conclusion

Vous devriez maintenant avoir une authentification basée sur une clé SSH configurée sur votre serveur, vous permettant de vous connecter sans fournir de mot de passe de compte.

# Comment installer Linux, Nginx, MariaDB, PHP (pile LEMP) sur Debian 10.

## Introduction.

La pile logicielle LEMP est un groupe de logiciels qui peuvent être utilisés pour servir des pages web dynamiques et des applications web. Le nom "LEMP" est un acronyme qui décrit un système d'exploitation **L**inux, avec un serveur web (**E**)Nginx. Les données sont stockées dans une base de données **M**ariaDB et le traitement dynamique est géré par **P**HP.

Bien que cette pile logicielle comprenne généralement MySQL comme système de gestion de base de données, certaines distributions Linux - dont Debian - utilisent MariaDB en remplacement de MySQL.

Dans ce guide, vous allez installer une pile LEMP sur un serveur Debian 10 en utilisant MariaDB comme système de gestion de base de données.

## Conditions préalables.

Pour compléter ce guide, vous devez avoir accès à un serveur Debian 10. Ce serveur doit avoir un utilisateur régulier configuré avec les privilèges sudo et un pare-feu activé avec ufw. Pour le configurer, vous pouvez suivre notre guide de configuration initiale du serveur avec Debian 10.

## Étape 1 - Installation du serveur web Nginx

Afin de servir les pages web aux visiteurs de votre site, nous allons utiliser Nginx, un serveur web populaire qui est bien connu pour ses performances et sa stabilité.

Tous les logiciels que vous utiliserez pour cette procédure proviendront directement des dépôts de paquets par défaut de Debian. Cela signifie que vous pouvez utiliser la suite de gestion de paquets apt pour compléter l'installation.

Comme c'est la première fois que vous utiliserez apt pour cette session, vous devriez commencer par mettre à jour votre index local de paquets. Vous pouvez ensuite installer le serveur :

$ sudo apt update

$ sudo apt install nginx

Sous Debian 10, Nginx est configuré pour commencer à fonctionner dès l'installation.

Si le pare-feu ufw fonctionne, vous devrez autoriser les connexions à Nginx. Vous devez activer le profil le plus restrictif qui permettra toujours le trafic que vous souhaitez. Comme vous n'avez pas encore configuré le protocole SSL pour votre serveur, il vous suffit pour l'instant d'autoriser le trafic HTTP sur le port 80.

Vous pouvez l'activer en tapant :

$ sudo ufw allow 'Nginx HTTP'

Vous pouvez vérifier le changement en tapant :

$ sudo ufw status

Vous devriez voir le trafic HTTP autorisé dans la sortie affichée :

Output  
Status: active  
  
To Action From  
-- ------ ----  
OpenSSH ALLOW Anywhere  
Nginx HTTP ALLOW Anywhere  
OpenSSH (v6) ALLOW Anywhere (v6)  
Nginx HTTP (v6) ALLOW Anywhere (v6)

Maintenant, vérifiez si le serveur est opérationnel en accédant au nom de domaine ou à l'adresse IP publique de votre serveur dans votre navigateur web. Si vous n'avez pas de nom de domaine pointé sur votre serveur et que vous ne connaissez pas l'adresse IP publique de votre serveur, vous pouvez la trouver en tapant l'un des éléments suivants dans votre terminal :

$ ip addr show eth0 | grep inet | awk '{ print $2; }' | sed 's/\/.\*$//'

Cela permettra d'imprimer quelques adresses IP. Vous pouvez essayer chacune d'entre elles à tour de rôle dans votre navigateur web.

Saisissez l'une des adresses que vous recevez dans votre navigateur web. Elle devrait vous amener à la page d'accueil par défaut de Nginx :

http://your\_domain\_or\_IP



Si vous voyez la page ci-dessus, c'est que vous avez installé Nginx avec succès.

## Étape 2 - Installation de MariaDB

Maintenant que vous avez un serveur web opérationnel, vous devez installer le système de base de données pour pouvoir stocker et gérer les données de votre site.

Dans la Debian 10, le méta-paquet mysql-server, qui était traditionnellement utilisé pour installer le serveur MySQL, a été remplacé par default-mysql-server,. Ce métapaquet fait référence à MariaDB, un fork communautaire du serveur MySQL original d'Oracle, et c'est actuellement le serveur de base de données par défaut compatible avec MySQL disponible sur les dépôts de gestionnaires de paquets basés sur debian.

Pour une compatibilité à plus long terme, il est toutefois recommandé d'installer MariaDB en utilisant le paquet actuel du programme, mariadb-server, au lieu d'utiliser le métapaquet.

Pour installer ce logiciel, exécutez :

$ sudo apt install mariadb-server

Une fois l'installation terminée, il est recommandé de lancer un script de sécurité préinstallé avec MariaDB. Ce script supprimera certains paramètres par défaut non sécurisés et verrouillera l'accès à votre système de base de données. Lancez le script interactif en exécutant :

$ sudo mysql\_secure\_installation

Ce script vous guidera à travers une série d'invites où vous pourrez apporter quelques modifications à votre configuration MariaDB. La première invite vous demandera d'entrer le **mot de passe root actuel** de la base de données. Il ne faut pas le confondre avec le **compte root** du système. **L'utilisateur root de la base de données** est un utilisateur administratif disposant de tous les privilèges sur le système de base de données. Comme vous venez d'installer MariaDB et que vous n'avez pas encore apporté de modifications à la configuration, ce mot de passe sera vide, il suffit donc d'appuyer sur la touche ENTER à l'invite.

L'invite suivante vous demande si vous souhaitez configurer un mot de passe pour **l'utilisateur root de la base de données**. Comme MariaDB utilise une méthode d'authentification spéciale pour **l'utilisateur root** qui est généralement plus sûre que l'utilisation d'un mot de passe, vous n'avez pas besoin de le définir maintenant. Tapez N puis appuyez sur ENTER.

De là, vous pouvez appuyer sur Y puis sur ENTER pour accepter les valeurs par défaut pour toutes les questions suivantes. Cela supprimera les utilisateurs anonymes et la base de données de test, désactivera la connexion à distance à la racine, et chargera ces nouvelles règles afin que MariaDB respecte immédiatement les modifications que vous avez apportées.

Lorsque vous avez terminé, connectez-vous à la console MariaDB en tapant :

$ sudo mariadb

Cela permettra de se connecter au serveur MariaDB en tant qu'utilisateur administratif **root** de la base de données, ce qui est déduit par l'utilisation de sudo lors de l'exécution de cette commande. Vous devriez voir le résultat comme ceci :

Output  
Welcome to the MariaDB monitor. Commands end with ; or \g.  
Your MariaDB connection id is 74  
Server version: 10.3.15-MariaDB-1 Debian 10  
  
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
MariaDB [(none)]>

Notez que vous n'avez pas eu besoin de fournir un mot de passe pour vous connecter en tant qu'**utilisateur root**. Cela fonctionne car la méthode d'authentification par défaut pour l'utilisateur administratif de MariaDB est unix-socket au lieu du mot de passe. Même si cela peut sembler être un problème de sécurité au premier abord, cela rend le serveur de base de données plus sûr car les seuls utilisateurs autorisés à se connecter en tant qu'**utilisateur MariaDB root** sont les utilisateurs système ayant des privilèges sudo qui se connectent depuis la console ou par le biais d'une application s'exécutant avec les mêmes privilèges. Concrètement, cela signifie que vous ne pourrez pas utiliser l'utilisateur **root** de la base de données pour vous connecter depuis votre application PHP.

Pour une sécurité accrue, il est préférable de créer des comptes d'utilisateurs dédiés avec des privilèges moins étendus pour chaque base de données, surtout si vous prévoyez d'héberger plusieurs bases de données sur votre serveur. Pour démontrer une telle configuration, nous allons créer une base de données nommée **example\_database** et un utilisateur nommé **example\_user**, mais vous pouvez remplacer ces noms par des valeurs différentes.

Pour créer une nouvelle base de données, exécutez la commande suivante depuis votre console MariaDB :

$ MariaDB [(none)]> CREATE DATABASE example\_database ;

# Essentiels SSH : Travailler avec les serveurs, les clients et les clés SSH.

## Introduction

SSH est un protocole sécurisé utilisé comme principal moyen de connexion à distance aux serveurs Linux. Il fournit une interface en mode texte en générant un shell distant. Après la connexion, toutes les commandes que vous tapez dans votre terminal local sont envoyées au serveur distant et y sont exécutées.

Dans ce guide de type "cheat sheet", nous allons couvrir quelques moyens courants de se connecter avec SSH pour atteindre vos objectifs. Il peut être utilisé comme référence rapide lorsque vous avez besoin de savoir comment vous connecter à votre serveur ou le configurer de différentes manières.

Comment utiliser ce guide

* Lisez d'abord la section " Vue d'ensemble de SSH " si vous ne connaissez pas SSH en général ou si vous commencez à peine.
* Utilisez les sections suivantes qui s'appliquent à ce que vous essayez de réaliser. La plupart des sections ne reposent sur aucune autre, vous pouvez donc utiliser les exemples suivants de manière indépendante.
* Copiez et collez les exemples donnés en ligne de commande, en remplaçant les valeurs mises en évidence par vos propres valeurs.

## Vue d'ensemble de SSH

La façon la plus courante de se connecter à un serveur Linux distant est par SSH. SSH signifie Secure Shell et fournit un moyen sûr et sécurisé d'exécuter des commandes, d'effectuer des modifications et de configurer des services à distance. Lorsque vous vous connectez via SSH, vous vous connectez en utilisant un compte qui existe sur le serveur distant.

### Comment fonctionne SSH

Lorsque vous vous connectez via SSH, vous vous retrouverez dans une session shell, qui est une interface en mode texte où vous pouvez interagir avec votre serveur. Pendant la durée de votre session SSH, toutes les commandes que vous tapez dans votre terminal local sont envoyées via un tunnel SSH crypté et exécutées sur votre serveur.

La connexion SSH est mise en œuvre selon un modèle client-serveur. Cela signifie que pour qu'une connexion SSH soit établie, la machine distante doit exécuter un logiciel appelé "daemon SSH". Ce logiciel écoute les connexions sur un port réseau spécifique, authentifie les demandes de connexion et génère l'environnement approprié si l'utilisateur fournit les bonnes informations d'identification.

L'ordinateur de l'utilisateur doit disposer d'un client SSH. Il s'agit d'un logiciel qui sait comment communiquer en utilisant le protocole SSH et qui peut recevoir des informations sur l'hôte distant auquel se connecter, le nom d'utilisateur à utiliser et les informations d'identification qui doivent être transmises pour s'authentifier. Le client peut également spécifier certains détails sur le type de connexion qu'il souhaite établir.

### Comment le SSH authentifie les utilisateurs

Les clients s'authentifient généralement à l'aide de mots de passe (moins sûrs et non recommandés) ou de clés SSH, qui sont très sûres.

Les mots de passe sont cryptés et sont faciles à comprendre pour les nouveaux utilisateurs. Toutefois, les robots et les utilisateurs malveillants tentent souvent à plusieurs reprises de s'authentifier sur des comptes qui autorisent les connexions par mot de passe, ce qui peut compromettre la sécurité. C'est pourquoi nous recommandons de toujours mettre en place une authentification basée sur une clé SSH pour la plupart des configurations.

Les clés SSH sont un ensemble de clés cryptographiques correspondantes qui peuvent être utilisées pour l'authentification. Chaque ensemble contient une clé publique et une clé privée. La clé publique peut être partagée librement sans inquiétude, tandis que la clé privée doit être gardée avec vigilance et ne jamais être exposée à quiconque.

Pour s'authentifier à l'aide de clés SSH, un utilisateur doit disposer d'une paire de clés SSH sur son ordinateur local. Sur le serveur distant, la clé publique doit être copiée dans un fichier situé dans le répertoire personnel de l'utilisateur, à l'adresse ~/.ssh/authorized\_keys. Ce fichier contient une liste de clés publiques, une par ligne, qui sont autorisées à se connecter à ce compte.

Lorsqu'un client se connecte à l'hôte, souhaitant utiliser l'authentification par clé SSH, il informe le serveur de cette intention et lui indique quelle clé publique utiliser. Le serveur vérifie alors la présence de la clé publique dans son fichier authorized\_keys, génère une chaîne aléatoire et la chiffre à l'aide de la clé publique. Ce message crypté ne peut être décrypté qu'avec la clé privée associée. Le serveur envoie ce message crypté au client pour vérifier s'il possède effectivement la clé privée associée.

À la réception de ce message, le client le décrypte en utilisant la clé privée et combine la chaîne aléatoire qui est révélée avec un identifiant de session négocié au préalable. Il génère ensuite un hachage MD5 de cette valeur et le transmet au serveur. Le serveur avait déjà le message original et l'ID de session, il peut donc comparer un hachage MD5 généré par ces valeurs et déterminer que le client doit avoir la clé privée.

Maintenant que vous savez comment SSH fonctionne, nous pouvons commencer à discuter de quelques exemples pour démontrer les différentes façons de travailler avec SSH

## Générer et travailler avec les clés SSH.

Cette section traitera de la manière de générer des clés SSH sur une machine cliente et de distribuer la clé publique aux serveurs où elles doivent être utilisées. C'est une bonne section pour commencer si vous n'avez pas encore généré de clés en raison de la sécurité accrue qu'elle permet pour les connexions futures.

### Générer une paire de clés SSH

La génération d'une nouvelle paire de clés publiques et privées SSH sur votre ordinateur local est la première étape vers l'authentification avec un serveur distant sans mot de passe. À moins qu'il n'y ait une bonne raison de ne pas le faire, vous devez toujours vous authentifier à l'aide de clés SSH.

Un certain nombre d'algorithmes cryptographiques peuvent être utilisés pour générer des clés SSH, notamment RSA, DSA et ECDSA. Les clés RSA sont généralement préférées et constituent le type de clé par défaut.

Pour générer une paire de clés RSA sur votre ordinateur local, tapez :

$ ssh-keygen

$ Generating public/private rsa key pair.  
Enter file in which to save the key (/home/demo/.ssh/id\_rsa):

Cette invite vous permet de choisir l'endroit où stocker votre clé privée RSA. Appuyez sur ENTER laisser cette option par défaut, ce qui les stockera dans le répertoire caché .ssh du répertoire d'origine de votre utilisateur. Laisser l'emplacement par défaut sélectionné permettra à votre client SSH de trouver les clés automatiquement.

L'invite suivante vous permet de saisir un mot de passe d'une longueur arbitraire pour sécuriser votre clé privée. Par défaut, vous devrez entrer le mot de passe que vous avez définie ici chaque fois que vous utiliserez la clé privée, par mesure de sécurité supplémentaire. N'hésitez pas à appuyer sur ENTER pour laisser cette case vide si vous ne voulez pas de mot de passe. Gardez cependant à l'esprit que cela permettra à toute personne qui prend le contrôle de votre clé privée de se connecter à vos serveurs.

Si vous choisissez d'entrer un mot de passe, rien ne sera affiché pendant que vous tapez. Il s'agit d'une mesure de sécurité.

Output  
Your identification has been saved in /root/.ssh/id\_rsa.  
Your public key has been saved in /root/.ssh/id\_rsa.pub.  
The key fingerprint is:  
8c:e9:7c:fa:bf:c4:e5:9c:c9:b8:60:1f:fe:1c:d3:8a root@here  
The key's randomart image is:  
+--[ RSA 2048]----+  
| |  
| |  
| |  
| + |  
| o S . |  
| o . \* + |  
| o + = O . |  
| + = = + |  
| ....Eo+ |  
+-----------------+

Cette procédure a généré une paire de clés SSH RSA, située dans le répertoire caché .ssh du répertoire personnel de votre utilisateur. Ces fichiers sont :

* ~/.ssh/id\_rsa : la clé privée. NE PARTAGEZ PAS CE FICHIER !
* ~/.ssh/id\_rsa.pub : La clé publique associée. Elle peut être partagée librement sans conséquence.

### Générer une paire de clés SSH avec un plus grand nombre de bits

Les clés SSH sont de 2048 bits par défaut. Cela est généralement considéré comme suffisant pour la sécurité, mais vous pouvez spécifier un plus grand nombre de bits pour une clé plus résistante.

Pour ce faire, incluez l'argument -b avec le nombre de bits que vous souhaitez. La plupart des serveurs prennent en charge les clés d'une longueur d'au moins 4096 bits. Les clés plus longues peuvent ne pas être acceptées pour la protection DDOS :

$ ssh-keygen -b 4096

Si vous aviez précédemment créé une clé différente, il vous sera demandé si vous souhaitez écraser votre clé précédente :

$ Overwrite (y/n)?

Si vous choisissez "yes", votre clé précédente sera écrasée et vous ne pourrez plus vous connecter aux serveurs en utilisant cette clé. Pour cette raison, veillez à écraser les clés avec précaution.

### Suppression ou modification de la phrase de passe d'une clé privée

Si vous avez généré un mot de passe pour votre clé privée et que vous souhaitez la modifier ou la supprimer, vous pouvez le faire facilement.

Note : Pour modifier ou supprimer le mot de passe, vous devez connaître le mot de passe original. Si vous avez perdu le mot de passe de la clé, il n'y a aucun recours et vous devrez générer une nouvelle paire de clés.

Pour modifier ou supprimer le mot de passe, il suffit de taper :

$ ssh-keygen -p

Enter file in which the key is (/root/.ssh/id\_rsa):

Vous pouvez taper l'emplacement de la touche que vous souhaitez modifier ou appuyer sur ENTER pour accepter la valeur par défaut :

Enter old passphrase:

Saisissez l'ancien mot de passe que vous souhaitez modifier. Un nouveau mot de passe vous sera alors demandé :

Enter new passphrase (empty for no passphrase):  
Enter same passphrase again:

Ici, entrez votre nouveau mot de passe ou appuyez sur la touche ENTER pour la supprimer.

### Affichage de l'empreinte de la clé SSH

Chaque paire de clés SSH partage une "empreinte" cryptographique unique qui peut être utilisée pour identifier les clés de manière unique. Cela peut être utile dans diverses situations.

Pour connaître l'empreinte d'une clé SSH, tapez :

$ ssh-keygen -l

Enter file in which the key is (/root/.ssh/id\_rsa):

Vous pouvez appuyer sur ENTER si c'est l'emplacement correct de la touche, sinon entrez l'emplacement révisé. Vous obtiendrez une chaîne contenant la longueur en bits de la clé, l'empreinte, le compte et l'hôte pour lesquels elle a été créée, ainsi que l'algorithme utilisé :

Output  
4096 8e:c4:82:47:87:c2:26:4b:68:ff:96:1a:39:62:9e:4e demo@test (RSA)

### Copier votre clé publique SSH sur un serveur avec SSH-Copy-ID

Pour copier votre clé publique sur un serveur, ce qui vous permet de vous authentifier sans mot de passe, plusieurs approches sont possibles.

Si vous avez actuellement un accès SSH par mot de passe configuré sur votre serveur, et que vous avez installé l'utilitaire ssh-copy-id, il s'agit d'un processus simple. L'outil ssh-copy-id est inclus dans les paquets OpenSSH de nombreuses distributions Linux, il est donc très probable qu'il soit installé par défaut.

Si vous disposez de cette option, vous pouvez facilement transférer votre clé publique en la tapant :

$ ssh-copy-id username@remote\_host

Le mot de passe du compte d'utilisateur vous sera alors demandé sur le système distant :

The authenticity of host '111.111.11.111 (111.111.11.111)' can't be established.  
ECDSA key fingerprint is fd:fd:d4:f9:77:fe:73:84:e1:55:00:ad:d6:6d:22:fe.  
Are you sure you want to continue connecting (yes/no)? Yes  
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed  
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys  
demo@111.111.11.111's password:

Après avoir saisi le mot de passe, le contenu de votre clé ~/.ssh/id\_rsa.pub sera ajouté à la fin du fichier ~/.ssh/authorized\_keys du compte utilisateur :

Output  
Number of key(s) added: 1  
  
Now try logging into the machine, with: "ssh 'demo@111.111.11.111'"  
and check to make sure that only the key(s) you wanted were added.

Vous pouvez maintenant vous connecter à ce compte sans mot de passe :

$ ssh username@remote\_host

### Copier votre clé publique SSH sur un serveur sans SSH-Copy-ID

Si vous ne disposez pas de l'utilitaire ssh-copy-id, mais que vous avez tout de même un accès SSH par mot de passe au serveur distant, vous pouvez copier le contenu de votre clé publique d'une autre manière.

Vous pouvez sortir le contenu de la clé et l'insérer dans la commande ssh. Sur le serveur distant, vous pouvez vous assurer que le répertoire ~/.ssh existe, puis ajouter le contenu de la clé au fichier ~/.ssh/authorized\_keys :

$ cat ~/.ssh/id\_rsa.pub | ssh username@remote\_host "mkdir -p ~/.ssh && cat >> ~/.ssh/authorized\_keys"

Il vous sera demandé de fournir le mot de passe du compte distant :

The authenticity of host '111.111.11.111 (111.111.11.111)' can't be established.  
ECDSA key fingerprint is fd:fd:d4:f9:77:fe:73:84:e1:55:00:ad:d6:6d:22:fe.  
Are you sure you want to continue connecting (yes/no)? Yes  
demo@111.111.11.111's password:

Après avoir saisi le mot de passe, votre clé sera copiée, ce qui vous permettra de vous connecter sans mot de passe :

$ ssh username@remote\_host

### Copie manuelle de votre clé publique SSH sur un serveur.

Si vous ne disposez pas d'un accès SSH avec mot de passe, vous devrez ajouter manuellement votre clé publique au serveur distant.

Sur votre machine locale, vous pouvez trouver le contenu de votre fichier de clé publique en tapant :

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

Il vous sera demandé de fournir le mot de passe du compte distant :

Output  
ssh-rsa  demo@test

Vous pouvez copier cette valeur, et la coller manuellement à l'endroit approprié sur le serveur distant. Vous devrez vous connecter au serveur distant par d'autres moyens.

Sur le serveur distant, créez le répertoire ~/.ssh s'il n'existe pas encore :

$ mkdir -p ~/.ssh

Ensuite, vous pouvez créer ou ajouter le fichier ~/.ssh/authorized\_keys en tapant :

$ echo public\_key\_string >> ~/.ssh/authorized\_keys

Vous devriez maintenant être en mesure de vous connecter au serveur distant sans mot de passe.

## Instructions de connexion de base

La section suivante aborde certaines des bases de la connexion à un serveur avec SSH.

### Connexion à un serveur distant

Pour vous connecter à un serveur distant et y ouvrir une session shell, vous pouvez utiliser la commande ssh.

La forme la plus simple suppose que votre nom d'utilisateur sur votre machine locale est le même que celui sur le serveur distant. Si c'est le cas, vous pouvez vous connecter en utilisant :

$ ssh remote\_host

Si votre nom d'utilisateur est différent sur le serveur distant, vous devez transmettre le nom de l'utilisateur distant comme ceci :

$ ssh username@remote\_host

La première fois que vous vous connectez à un nouvel hôte, vous verrez un message qui ressemble à ceci :

The authenticity of host '111.111.11.111 (111.111.11.111)' can't be established.  
ECDSA key fingerprint is fd:fd:d4:f9:77:fe:73:84:e1:55:00:ad:d6:6d:22:fe.  
Are you sure you want to continue connecting (yes/no)? yes

Tapez "yes" pour accepter l'authenticité de l'hôte distant.

Si vous utilisez l'authentification par mot de passe, le mot de passe du compte distant vous sera demandé ici. Si vous utilisez des clés SSH, vous serez invité à saisir la phrase de passe de votre clé privée si elle est définie, sinon vous serez automatiquement connecté.

### Exécution d'une commande unique sur un serveur distant

Pour exécuter une commande unique sur un serveur distant au lieu de créer une session shell, vous pouvez ajouter la commande après les informations de connexion, comme ceci :

$ ssh username@remote\_host command\_to\_run

Cela permettra de se connecter à l'hôte distant, de s'authentifier avec vos identifiants et d'exécuter la commande que vous avez spécifiée. La connexion sera immédiatement fermée par la suite.

### Connexion à un serveur avec un port différent

Par défaut, le démon SSH d'un serveur fonctionne sur le port 22. Votre client SSH supposera que c'est le cas lorsqu'il essaiera de se connecter. Si votre serveur SSH écoute sur un port non standard (ceci est démontré dans une section ultérieure), vous devrez spécifier le nouveau numéro de port lors de la connexion avec votre client.

Vous pouvez le faire en spécifiant le numéro de port avec l'option -p :

$ ssh -p port\_num username@remote\_host

Pour éviter d'avoir à le faire chaque fois que vous vous connectez à votre serveur distant, vous pouvez créer ou modifier un fichier de configuration dans le répertoire ~/.ssh du répertoire d'accueil de votre ordinateur local.

Modifiez ou créez le fichier maintenant en tapant :

$ nano ~/.ssh/config

Ici, vous pouvez définir des options de configuration spécifiques à l'hôte. Pour spécifier votre nouveau port, utilisez un format comme celui-ci :

Host remote\_alias  
 HostName remote\_host  
 Port port\_num

Cela vous permettra de vous connecter sans avoir à spécifier le numéro de port spécifique sur la ligne de commande.

### Ajout de vos clés SSH à un agent SSH pour éviter de taper le mot de passe

Si votre clé privée SSH comporte un mot de passe, vous serez invité à la saisir chaque fois que vous l'utiliserez pour vous connecter à un hôte distant.

Pour éviter d'avoir à le faire de manière répétée, vous pouvez lancer un agent SSH. Ce petit utilitaire stocke votre clé privée après que vous ayez saisi le mot de passe pour la première fois. Il sera disponible pendant toute la durée de votre session de terminal, ce qui vous permettra de vous connecter à l'avenir sans avoir à saisir à nouveau le mot de passe.

Ceci est également important si vous devez transmettre vos identifiants SSH (voir plus loin).

Pour lancer l'agent SSH, tapez ce qui suit dans votre session de terminal locale :

$ eval $(ssh-agent)

Output  
Agent pid 10891

Cela permettra de lancer le programme d'agent et de le placer en arrière-plan. Maintenant, vous devez ajouter votre clé privée à l'agent, afin qu'il puisse gérer votre clé :

$ ssh-add

Enter passphrase for /home/demo/.ssh/id\_rsa:  
Identity added: /home/demo/.ssh/id\_rsa (/home/demo/.ssh/id\_rsa)

De là, vous pouvez entrer en SSH dans tout autre hôte auquel votre clé SSH est autorisée à accéder. Vous vous connecterez comme si votre clé SSH privée était située sur ce serveur.

## Options de configuration côté serveur

Cette section contient quelques options de configuration courantes côté serveur qui peuvent déterminer la manière dont votre serveur répond et les types de connexions autorisées.

### Désactivation de l'authentification par mot de passe

Si vous avez des clés SSH configurées, testées et fonctionnant correctement, c'est probablement une bonne idée de désactiver l'authentification par mot de passe. Cela empêchera tout utilisateur de se connecter avec SSH en utilisant un mot de passe.

Pour ce faire, connectez-vous à votre serveur distant et ouvrez le fichier /etc/ssh/sshd\_config avec les privilèges root ou sudo :

$ sudo nano /etc/ssh/sshd\_config

A l'intérieur du fichier, recherchez la directive PasswordAuthentication. Si elle est commentée, décommentez-la. Mettez la valeur "no" pour désactiver l'authentification par mot de passe :

PasswordAuthentication no

Après avoir effectué la modification, enregistrez et fermez le fichier. Pour mettre en œuvre les changements, vous devez redémarrer le service SSH.

$ sudo service ssh restart

Désormais, tous les comptes du système ne pourront plus se connecter avec SSH à l'aide de mots de passe.

### Changer le port sur lequel tourne le daemon SSH

Certains administrateurs vous suggèrent de changer le port par défaut sur lequel tourne SSH. Cela peut contribuer à réduire le nombre de tentatives d'authentification de votre serveur par des robots automatisés.

Pour changer le port sur lequel le démon SSH écoute, vous devrez vous connecter à votre serveur distant. Ouvrez le fichier sshd\_config sur le système distant avec les privilèges de root, soit en vous connectant avec cet utilisateur, soit en utilisant sudo :

$ sudo nano /etc/ssh/sshd\_config

Une fois à l'intérieur, vous pouvez changer le port sur lequel fonctionne SSH en trouvant la spécification Port 22 et en la modifiant pour refléter le port que vous souhaitez utiliser. Par exemple, pour changer le port en 4444, mettez ceci dans votre fichier :

#Port 22  
Port 4444

Enregistrez et fermez le fichier lorsque vous avez terminé. Pour mettre en œuvre les changements, vous devez redémarrer le démon SSH.

$ sudo service ssh restart

Après le redémarrage du daemon, vous devrez vous authentifier en spécifiant le numéro de port (démontré dans une section précédente).

### Limiter les utilisateurs qui peuvent se connecter par le biais des sciences humaines

Pour limiter explicitement les comptes d'utilisateurs qui peuvent se connecter via SSH, vous pouvez adopter quelques approches différentes, chacune d'entre elles impliquant l'édition du fichier de configuration du démon SSH.

Sur votre serveur distant, ouvrez ce fichier maintenant avec les privilèges root ou sudo :

$ sudo nano /etc/ssh/sshd\_config

La première méthode pour spécifier les comptes qui sont autorisés à se connecter est d'utiliser la directive AllowUsers. Recherchez la directive AllowUsers dans le fichier. Si elle n'existe pas, créez-la n'importe où. Après la directive, dressez la liste des comptes d'utilisateurs qui doivent être autorisés à se connecter en SSH :

AllowUsers user1 user2

Enregistrez et fermez le fichier. Redémarrez le démon pour implémenter vos modifications.

$ sudo service ssh restart

Si vous êtes plus à l'aise avec la gestion de groupe, vous pouvez utiliser la directive AllowGroups à la place. Si c'est le cas, ajoutez simplement un seul groupe qui devrait être autorisé à accéder aux SSH (nous allons créer ce groupe et ajouter des membres momentanément) :

AllowGroups sshmembers

Enregistrez et fermez le fichier.

Maintenant, vous pouvez créer un groupe système (sans répertoire d'origine) correspondant au groupe que vous avez spécifié en tapant :

$ sudo groupadd -r sshmembers

Veillez à ajouter à ce groupe tous les comptes d'utilisateurs dont vous avez besoin. Cela peut être fait en tapant :

$ sudo usermod -a -G sshmembers user1  
$ sudo usermod -a -G sshmembers user2

Maintenant, redémarrez le démon SSH pour mettre en œuvre vos changements.

$ sudo service ssh restart

### Désactivation de la connexion root

Il est souvent conseillé de désactiver complètement la connexion root via SSH après avoir créé un compte utilisateur SSH qui a des privilèges sudo.

Pour ce faire, ouvrez le fichier de configuration du démon SSH avec root ou sudo sur votre serveur distant.

$ sudo nano /etc/ssh/sshd\_config

A l'intérieur, cherchez la directive PermitRootLogin. Si elle est commentée, décommentez-la. Modifiez la valeur à "no" :

PermitRootLogin no

Enregistrez et fermez le fichier. Pour mettre en œuvre vos changements, redémarrez le daemon SSH.

$ sudo service ssh restart

### Autoriser l'accès root pour des commandes spécifiques

Dans certains cas, il est possible de désactiver l'accès à la racine de manière générale, mais de l'activer afin de permettre à certaines applications de fonctionner correctement. Il peut s'agir, par exemple, d'une routine de sauvegarde.

Cela peut être réalisé par le biais du fichier authorized\_keys de l'utilisateur root, qui contient les clés SSH autorisées à utiliser le compte.

Ajoutez la clé de votre ordinateur local que vous souhaitez utiliser pour ce processus (nous recommandons de créer une nouvelle clé pour chaque processus automatique) au fichier authorized\_keys de l'utilisateur root sur le serveur. Nous allons faire une démonstration avec la commande ssh-copy-id ici, mais vous pouvez utiliser n'importe laquelle des méthodes de copie de clés dont nous parlons dans d'autres sections :

$ ssh-copy-id root@remote\_host

Maintenant, connectez-vous au serveur distant. Nous devrons ajuster l'entrée dans le fichier authorized\_keys, donc l'ouvrir avec un accès root ou sudo :

$ sudo nano /root/.ssh/authorized\_keys

Au début de la ligne contenant la clé que vous avez téléchargée, ajoutez une liste command= qui définit la commande pour laquelle cette clé est valable. Cette liste doit inclure le chemin d'accès complet à l'exécutable, ainsi que les arguments éventuels :

command="/path/to/command arg1 arg2" ssh-rsa ...

Enregistrez et fermez le fichier lorsque vous avez terminé.

Maintenant, ouvrez le fichier sshd\_config avec les privilèges root ou sudo :

$ sudo nano /etc/ssh/sshd\_config

Trouvez la directive PermitRootLogin, et changez la valeur en forced-commands-only. Cela ne permettra aux connexions de clés SSH d'utiliser root que lorsqu'une commande a été spécifiée pour la clé :

PermitRootLogin forced-commands-only

Enregistrez et fermez le fichier. Redémarrez le daemon SSH pour implémenter vos modifications.

$ sudo service ssh restart

### Transmission de l’affichage des applications X au client.

Le démon SSH peut être configuré pour transférer automatiquement l'affichage des applications X sur le serveur vers la machine cliente. Pour que cela fonctionne correctement, le client doit avoir un système X windows configuré et activé.

Pour activer cette fonctionnalité, connectez-vous à votre serveur distant et modifiez le fichier sshd\_config en tant que root ou avec les privilèges sudo :

$ sudo nano /etc/ssh/sshd\_config

Recherchez la directive X11Forwarding. Si elle est commentée, décommentez-la. Créez-la si nécessaire et mettez la valeur à "yes" :

X11Forwarding yes

Enregistrez et fermez le fichier. Redémarrez le daemon SSH pour implémenter vos modifications.

$ sudo service ssh restart

Les applications graphiques lancées sur le serveur au cours de cette session doivent être affichées sur l'ordinateur local. La performance peut être un peu lente, mais elle est très utile en cas de problème.

## Options de configuration côté client

Dans la prochaine section, nous nous concentrerons sur certains ajustements que vous pouvez faire du côté client de la connexion.

### Définition des informations de connexion spécifiques au serveur

Sur votre ordinateur local, vous pouvez définir des configurations individuelles pour tout ou partie des serveurs auxquels vous vous connectez. Ces configurations peuvent être stockées dans le fichier ~/.ssh/config, qui est lu par votre client SSH à chaque appel.

Créez ou ouvrez ce fichier dans votre éditeur de texte sur votre ordinateur local :

$ nano ~/.ssh/config

À l'intérieur, vous pouvez définir des options de configuration individuelles en introduisant chacune d'entre elles par un mot-clé Host, suivi d'un alias. En dessous et en retrait, vous pouvez définir n'importe laquelle des directives trouvées dans la page de manuel ssh\_config :

Host testhost  
 HostName your\_domain  
 Port 4444  
 User demo

Vous pourrez alors vous connecter à your\_domain sur le port 4444 en utilisant le nom d'utilisateur demo en tapant simplement :

$ ssh testhost

Vous pouvez également utiliser des jokers pour correspondre à plusieurs hôtes. N'oubliez pas que les correspondances ultérieures peuvent remplacer les précédentes. C'est pourquoi vous devez placer vos correspondances les plus générales en tête de liste. Par exemple, vous pouvez faire en sorte que toutes les connexions ne permettent pas la redirection X par défaut, avec une surcharge pour your\_domain en l'indiquant dans votre fichier :

Host \*  
 ForwardX11 no  
  
Host testhost  
 HostName your\_domain  
 ForwardX11 yes  
 Port 4444  
 User demo

Enregistrez et fermez le fichier lorsque vous avez terminé.

### Maintenir les connexions pour éviter les interruptions

Si vous vous retrouvez déconnecté des sessions SSH avant d'être prêt, il est possible que votre connexion soit interrompue.

Vous pouvez configurer votre client pour qu'il envoie un paquet au serveur de temps en temps afin d'éviter cette situation :

Sur votre ordinateur local, vous pouvez configurer cela pour chaque connexion en éditant votre fichier ~/.ssh/config. Ouvrez-le maintenant :

$ nano ~/.ssh/config

Si elle n'existe pas encore, définissez en haut du fichier une section qui correspondra à tous les hôtes. Réglez le ServerAliveInterval sur "120" pour envoyer un paquet au serveur toutes les deux minutes. Cela devrait être suffisant pour avertir le serveur de ne pas fermer la connexion :

Host \*  
 ServerAliveInterval 120

Enregistrez et fermez le fichier lorsque vous avez terminé.

### Désactivation du contrôle de l'hôte

Par défaut, chaque fois que vous vous connectez à un nouveau serveur, l'empreinte de la clé hôte du daemon SSH distant s'affiche.

The authenticity of host '111.111.11.111 (111.111.11.111)' can't be established.  
ECDSA key fingerprint is fd:fd:d4:f9:77:fe:73:84:e1:55:00:ad:d6:6d:22:fe.  
Are you sure you want to continue connecting (yes/no)? yes

Il est configuré de manière à ce que vous puissiez vérifier l'authenticité de l'hôte auquel vous tentez de vous connecter et repérer les cas où un utilisateur malveillant pourrait tenter de se faire passer pour l'hôte distant.

Dans certaines circonstances, vous pouvez souhaiter désactiver cette fonction. Note : Cela peut représenter un risque de sécurité important, alors assurez-vous de savoir ce que vous faites si vous configurez votre système de cette manière.

Pour effectuer le changement, ouvrez le fichier ~/.ssh/config sur votre ordinateur local :

$ nano ~/.ssh/config

Si elle n'existe pas encore, définissez en haut du fichier une section qui correspondra à tous les hôtes. Définissez la directive StrictHostKeyChecking à no pour ajouter automatiquement de nouveaux hôtes au fichier known\_hosts. Définissez la directive UserKnownHostsFile à /dev/null pour ne pas être averti sur les hôtes nouveaux ou modifiés :

Host \*  
 StrictHostKeyChecking no  
 UserKnownHostsFile /dev/null

Vous pouvez activer la vérification au cas par cas en inversant ces options pour les autres hôtes. La valeur par défaut de StrictHostKeyChecking est ask :

Host \*  
 StrictHostKeyChecking no  
 UserKnownHostsFile /dev/null  
  
Host testhost  
 HostName your\_domain  
 StrictHostKeyChecking ask  
 UserKnownHostsFile /home/demo/.ssh/known\_hosts

### Multiplexage de SSH sur une seule connexion TCP

Dans certaines situations, l'établissement d'une nouvelle connexion TCP peut prendre plus de temps que vous ne le souhaiteriez. Si vous établissez plusieurs connexions à la même machine, vous pouvez profiter du multiplexage.

Le multiplexage SSH réutilise la même connexion TCP pour plusieurs sessions SSH. Cela supprime une partie du travail nécessaire à l'établissement d'une nouvelle session, ce qui peut accélérer les choses. Limiter le nombre de connexions peut également être utile pour d'autres raisons.

Pour mettre en place le multiplexage, vous pouvez configurer manuellement les connexions, ou vous pouvez configurer votre client pour qu'il utilise automatiquement le multiplexage lorsqu'il est disponible. Nous allons démontrer la deuxième option ici.

Pour configurer le multiplexage, modifiez le fichier de configuration de votre client SSH sur votre machine locale :

$ nano ~/.ssh/config

Si vous n'avez pas encore de définition d'hôte joker en haut du fichier, ajoutez-en une maintenant (en tant qu'hôte \*). Nous allons définir les valeurs ControlMaster, ControlPath et ControlPersist pour établir notre configuration de multiplexage.

Le ControlMaster doit être réglé sur auto pour pouvoir autoriser automatiquement le multiplexage si possible. Le ControlPath établira le chemin vers le socket de contrôle. La première session créera ce socket et les sessions suivantes pourront le trouver car il est étiqueté par nom d'utilisateur, hôte et port.

En réglant l'option ControlPersist sur 1, la connexion maître initiale pourra être établie en arrière-plan. Le 1 spécifie que la connexion TCP doit se terminer automatiquement une seconde après la fermeture de la dernière session SSH :

Host \*  
 ControlMaster auto  
 ControlPath ~/.ssh/multiplex/%r@%h:%p  
 ControlPersist 1

Enregistrez et fermez le fichier lorsque vous avez terminé. Maintenant, nous devons créer le répertoire que nous avons spécifié dans le chemin de contrôle :

$ mkdir ~/.ssh/multiplex

Désormais, toute session établie avec la même machine tentera d'utiliser le socket et la connexion TCP existants. Lorsque la dernière session existe, la connexion est interrompue au bout d'une seconde.

Si, pour une raison quelconque, vous devez contourner temporairement la configuration de multiplexage, vous pouvez le faire en passant le drapeau -S à none :

$ ssh -S none username@remote\_host

## Mise en place des tunnels SSH

Faire passer d'autres trafics par un tunnel SSH sécurisé est un excellent moyen de contourner les paramètres restrictifs du pare-feu. C'est également un excellent moyen de chiffrer le trafic réseau qui, autrement, ne serait pas chiffré.

### Configuration d'un tunnel local vers un serveur

Les connexions SSH peuvent être utilisées pour faire passer le trafic des ports de l'hôte local aux ports d'un hôte distant.

Une connexion locale est un moyen d'accéder à un emplacement du réseau à partir de votre ordinateur local par l'intermédiaire de votre hôte distant. Tout d'abord, une connexion SSH est établie avec votre hôte distant. Sur le serveur distant, une connexion est établie à une adresse réseau externe (ou interne) fournie par l'utilisateur et le trafic vers cet emplacement est tunnelisé vers votre ordinateur local sur un port spécifié.

Cette méthode est souvent utilisée pour établir un tunnel vers un environnement réseau moins restreint en contournant un pare-feu. Une autre utilisation courante est l'accès à une interface web "localhost" à partir d'un site distant.

Pour établir un tunnel local vers votre serveur distant, vous devez utiliser le paramètre -L lors de la connexion et vous devez fournir trois informations supplémentaires :

* Le port local où vous souhaitez accéder à la connexion par tunnel.
* L'hôte auquel vous souhaitez que votre hôte distant se connecte.
* Le port sur lequel vous souhaitez que votre hôte distant se connecte.

Ces informations sont données, dans l'ordre ci-dessus (séparées par des deux-points), en tant qu'arguments du drapeau -L. Nous utiliserons également le drapeau -f, qui fait que SSH passe en arrière-plan avant de s'exécuter et le drapeau -N, qui n'ouvre pas de shell ou n'exécute pas de programme du côté distant.

Par exemple, pour vous connecter à your\_domain sur le port 80 de votre hôte distant, en rendant la connexion disponible sur votre machine locale sur le port 8888, vous pourriez taper

$ ssh -f -N -L 8888:your\_domain:80 username@remote\_host

Maintenant, si vous faites pointer votre navigateur web local sur 127.0.0.1:8888, vous devriez voir tout le contenu de your\_domain sur le port 80.

Voici un guide plus général de la syntaxe :

$ ssh -L your\_port:site\_or\_IP\_to\_access:site\_port username@host

Comme la connexion est en arrière-plan, vous devrez trouver son PID pour la tuer. Vous pouvez le faire en recherchant le port que vous avez transféré :

$ ps aux | grep 8888

Output  
1001 5965 0.0 0.0 48168 1136 ? Ss 12:28 0:00 ssh -f -N -L 8888:your\_domain:80 username@remote\_host  
1001 6113 0.0 0.0 13648 952 pts/2 S+ 12:37 0:00 grep --colour=auto 8888

Vous pouvez alors tuer le processus en ciblant le PID, qui est le numéro dans la deuxième colonne de la ligne qui correspond à votre commande SSH :

$ kill 5965

Une autre option consiste à démarrer la connexion sans le drapeau -f. La connexion restera alors au premier plan, ce qui vous empêchera d'utiliser la fenêtre du terminal pendant toute la durée de la transmission. L'avantage de cette option est que vous pouvez facilement tuer le tunnel en tapant CTRL-C.

### Configurer le tunnelage à distance vers un serveur

Les connexions SSH peuvent être utilisées pour faire passer le trafic des ports de l'hôte local aux ports d'un hôte distant.

Dans un tunnel distant, une connexion est établie vers un hôte distant. Lors de la création du tunnel, un port distant est spécifié. Ce port, sur l'hôte distant, sera ensuite tunnelisé vers une combinaison d'hôte et de port à laquelle est connecté l'ordinateur local. Cela permettra à l'ordinateur distant d'accéder à un hôte par l'intermédiaire de votre ordinateur local.

Cela peut être utile si vous devez autoriser l'accès à un réseau interne qui est verrouillé aux connexions externes. Si le pare-feu autorise les connexions en dehors du réseau, cela vous permettra de vous connecter à une machine distante et de canaliser le trafic de cette machine vers un endroit du réseau interne.

Pour établir un tunnel distant vers votre serveur distant, vous devez utiliser le paramètre -R lors de la connexion et fournir trois informations supplémentaires :

* Le port par lequel l'hôte distant peut accéder à la connexion par tunnel.
* L'hôte auquel vous voulez que votre ordinateur local se connecte.
* Le port auquel vous voulez que votre ordinateur local se connecte.

Ces informations sont données, dans l'ordre ci-dessus (séparées par des deux-points), comme arguments du drapeau -R. Nous utiliserons également le drapeau -f, qui fait que SSH passe en arrière-plan avant de s'exécuter et le drapeau -N, qui n'ouvre pas de shell ou n'exécute pas de programme du côté distant.

Par exemple, pour vous connecter à your\_domain sur le port 80 de notre ordinateur local, rendant la connexion disponible sur notre hôte distant sur le port 8888, vous pourriez taper :

$ ssh -f -N -R 8888:your\_domain:80 username@remote\_host

Maintenant, sur l'hôte distant, ouvrir un navigateur web au 127.0.0.1:8888 vous permettrait de voir tout le contenu de your\_domain sur le port 80.

Voici un guide plus général de la syntaxe :

$ ssh -R remote\_port:site\_or\_IP\_to\_access:site\_port username@host

Comme la connexion est en arrière-plan, vous devrez trouver son PID pour la tuer. Vous pouvez le faire en recherchant le port que vous avez transféré :

$ ps aux | grep 8888

Output  
1001 5965 0.0 0.0 48168 1136 ? Ss 12:28 0:00 ssh -f -N -R 8888:your\_domain:80 username@remote\_host  
1001 6113 0.0 0.0 13648 952 pts/2 S+ 12:37 0:00 grep --colour=auto 8888

Vous pouvez alors tuer le processus en ciblant le PID, qui est le numéro dans la deuxième colonne, de la ligne qui correspond à votre commande SSH :

$ kill 5965

Une autre option consiste à démarrer la connexion sans le drapeau -f. La connexion restera alors au premier plan, ce qui vous empêchera d'utiliser la fenêtre du terminal pendant toute la durée de la transmission. L'avantage de cette option est que vous pouvez facilement tuer le tunnel en tapant CTRL-C.

### Configuration d'un tunnel dynamique vers un serveur distant

Les connexions SSH peuvent être utilisées pour faire passer le trafic des ports de l'hôte local aux ports d'un hôte distant.

Un tunnel dynamique est similaire à un tunnel local en ce sens qu'il permet à l'ordinateur local de se connecter à d'autres ressources via un hôte distant. Un tunnel dynamique fait cela en spécifiant simplement un seul port local. Les applications qui souhaitent tirer parti de ce port pour le tunnelage doivent pouvoir communiquer en utilisant le protocole SOCKS afin que les paquets puissent être correctement redirigés de l'autre côté du tunnel.

Le trafic qui passe par ce port local sera envoyé à l'hôte distant. De là, le protocole SOCKS sera interprété pour établir une connexion vers l'emplacement final souhaité. Cette configuration permet à une application compatible SOCKS de se connecter à un nombre quelconque d'emplacements par l'intermédiaire du serveur distant, sans avoir à passer par de multiples tunnels statiques.

Pour établir la connexion, nous passerons le drapeau -D avec le port local où nous souhaitons accéder au tunnel. Nous utiliserons également le drapeau -f, qui fait que SSH passe en arrière-plan avant de s'exécuter et le drapeau-N, qui n'ouvre pas de shell ou n'exécute pas de programme du côté distant.

Par exemple, pour établir un tunnel sur le port 7777, vous pouvez taper :

$ ssh -f -N -D 7777 username@remote\_host

À partir de là, vous pouvez commencer à pointer votre application compatible SOCKS (comme un navigateur web), vers le port que vous avez sélectionné. L'application enverra ses informations dans une socket associée au port.

La méthode pour diriger le trafic vers le port SOCKS diffère selon l'application. Par exemple, dans Firefox, l'emplacement général est Préférences > Avancé > Paramètres > Configurations manuelles du proxy. Dans Chrome, vous pouvez démarrer l'application avec le drapeau --proxy-server=.activé. Vous voudrez utiliser l'interface localhost et le port que vous avez transféré.

Comme la connexion est en arrière-plan, vous devrez trouver son PID pour la tuer. Vous pouvez le faire en recherchant le port que vous avez transféré :

$ ps aux | grep 8888

Output  
1001 5965 0.0 0.0 48168 1136 ? Ss 12:28 0:00 ssh -f -N -L 8888:your\_domain:80 username@remote\_host  
1001 6113 0.0 0.0 13648 952 pts/2 S+ 12:37 0:00 grep --colour=auto 8888

Vous pouvez alors tuer le processus en ciblant le PID, qui est le numéro dans la deuxième colonne, de la ligne qui correspond à votre commande SSH :

$ kill 5965

Une autre option consiste à démarrer la connexion sans le drapeau -f. La connexion restera alors au premier plan, ce qui vous empêchera d'utiliser la fenêtre du terminal pendant toute la durée de la transmission. L'avantage de cette option est que vous pouvez facilement tuer le tunnel en tapant CTRL-C.

## Utilisation des codes d'échappement SSH pour contrôler les connexions

Même après avoir établi une session SSH, il est possible d'exercer un contrôle sur la connexion depuis le terminal. Nous pouvons le faire grâce à des codes d'échappement SSH, qui nous permettent d'interagir avec notre logiciel SSH local au cours d'une session.

### Forcer une déconnexion du côté client (Comment sortir d'une session bloquée ou gelée)

Une des caractéristiques les plus utiles d'OpenSSH qui passe largement inaperçue est la possibilité de contrôler certains aspects de la session de l'intérieur.

Ces commandes peuvent être exécutées en commençant par le caractère ~ de contrôle au sein d'une session SSH. Les commandes de contrôle ne seront interprétées que si elles sont la première chose qui est tapée après un saut de ligne, donc toujours appuyer sur ENTER une ou deux fois avant d'en utiliser une.

L'une des commandes les plus utiles est la possibilité d'initier une déconnexion du client. Les connexions SSH sont généralement fermées par le serveur, mais cela peut poser un problème si le serveur souffre de problèmes ou si la connexion a été interrompue. En utilisant une déconnexion côté client, la connexion peut être proprement fermée à partir du client.

Pour fermer une connexion depuis le client, utilisez le caractère de contrôle (~), avec un point. Si votre connexion rencontre des problèmes, vous serez probablement dans ce qui semble être une session de terminal bloquée. Tapez les commandes malgré l'absence de retour d'information pour effectuer une déconnexion côté client :

$ [ENTER]  
$ ~.

La connexion devrait se fermer immédiatement, vous ramenant à votre session shell locale.

### Placer une session SSH en arrière-plan

Une des caractéristiques les plus utiles d'OpenSSH qui passe largement inaperçue est la possibilité de contrôler certains aspects de la session depuis la connexion.

Ces commandes peuvent être exécutées en commençant par le caractère ~ de contrôle depuis une connexion SSH. Les commandes de contrôle ne seront interprétées que si elles sont la première chose qui est tapée après une nouvelle ligne, donc toujours appuyer sur ENTER une ou deux fois avant d'en utiliser une.

Cela permet notamment de mettre une session SSH en arrière-plan. Pour ce faire, nous devons fournir le caractère de contrôle (~) et ensuite exécuter le raccourci clavier conventionnel pour mettre une tâche en arrière-plan (CTRL-z) :

$ [ENTER]  
$ ~[CTRL-z]

La connexion sera alors placée en arrière-plan, ce qui vous ramènera à votre session shell locale. Pour revenir à votre session SSH, vous pouvez utiliser les mécanismes classiques de contrôle de l'emploi.

Vous pouvez immédiatement réactiver votre tâche la plus récente en arrière-plan en tapant :

$ fg

Si vous avez plusieurs tâches en arrière-plan, vous pouvez voir les emplois disponibles en tapant :

$ jobs

Output  
[1]+ Stopped ssh username@some\_host  
[2] Stopped ssh username@another\_host

Vous pouvez ensuite mettre en avant n'importe quelle tâche en utilisant l'index dans la première colonne avec un signe de pourcentage :

$ jg %2

### Modification des options de transfert de port sur une connexion SSH existante

Une des caractéristiques les plus utiles d'OpenSSH qui passe largement inaperçue est la possibilité de contrôler certains aspects de la session depuis la connexion.

Ces commandes peuvent être exécutées en commençant par le caractère ~ de contrôle depuis une connexion SSH. Les commandes de contrôle ne seront interprétées que si elles sont la première chose qui est tapée après une nouvelle ligne, donc toujours appuyer sur ENTER une ou deux fois avant d'en utiliser une.

Cela permet notamment à l'utilisateur de modifier la configuration de la redirection de port après que la connexion ait déjà été établie. Cela permet de créer ou de supprimer des règles de redirection de port à la volée.

Ces fonctionnalités font partie de l'interface de ligne de commande SSH, à laquelle on peut accéder pendant une session en utilisant le caractère de contrôle (~) et le "C" :

$ [ENTER]  
$ ~C

ssh>

Vous recevrez une invite de commande SSH, qui comporte un ensemble très limité de commandes valides. Pour voir les options disponibles, vous pouvez taper -h à partir de cette invite. Si rien n'est renvoyé, vous devrez peut-être augmenter la verbosité de votre sortie SSH en utilisant ~v plusieurs fois :

$ [ENTER]  
$ ~v  
$ ~v  
$ ~v

$ ~C  
$ -h

Commands:  
 -L[bind\_address:]port:host:hostport Request local forward  
 -R[bind\_address:]port:host:hostport Request remote forward  
 -D[bind\_address:]port Request dynamic forward  
 -KL[bind\_address:]port Cancel local forward  
 -KR[bind\_address:]port Cancel remote forward  
 -KD[bind\_address:]port Cancel dynamic forward

Comme vous pouvez le voir, vous pouvez facilement mettre en œuvre n'importe quelle option de transmission en utilisant les options appropriées (voir la section sur la transmission pour plus d'informations). Vous pouvez également détruire un tunnel avec la commande "kill" associée, indiquée par un "K" avant la lettre de type de transfert. Par exemple, pour tuer une redirection locale (-L), vous pouvez utiliser la commande -KL. Il vous suffira de fournir le port pour cela.

Ainsi, pour mettre en place une redirection locale, vous pouvez taper :

$ [ENTER]  
$ ~C  
$ -L 8888:127.0.0.1:80

Le port 8888 de votre ordinateur local pourra désormais communiquer avec le serveur web de l'hôte auquel vous vous connectez. Lorsque vous avez terminé, vous pouvez démonter ce transfert en tapant :

$ [ENTER]  
$ ~C  
$ -KL 8888

## Conclusion

Les instructions ci-dessus devraient couvrir la majorité des informations dont la plupart des utilisateurs auront besoin au quotidien sur fondamentaux de SSH.

# UFW Essentials : Règles et commandes communes pour les pares-feux

## Introduction

UFW est un outil de configuration de pare-feu pour iptables qui est inclus par défaut avec Ubuntu. Ce guide de type "cheat sheet" fournit une référence rapide aux commandes UFW qui permettront de créer des règles de pare-feu pour iptables, utiles dans des scénarios courants et quotidiens. Il comprend des exemples UFW d'autorisation et de blocage de divers services par port, interface réseau et adresse IP source.

## Comment utiliser ce guide

Si vous commencez tout juste à utiliser l'UFW pour configurer votre pare-feu, consultez notre introduction à l'UFW

La plupart des règles qui sont décrites ici supposent que vous utilisez le jeu de règles UFW par défaut. C'est-à-dire qu'il est défini pour autoriser le trafic sortant et refuser le trafic entrant, par le biais des règles par défaut, de sorte que vous devez autoriser sélectivement le trafic entrant

Utilisez les sections suivantes qui sont applicables à ce que vous essayez de réaliser. La plupart des sections ne reposent sur aucune autre, vous pouvez donc utiliser les exemples ci-dessous de manière indépendante

Utilisez le menu Contenu sur le côté droit de cette page (en grande largeur) ou la fonction de recherche de votre navigateur pour localiser les sections dont vous avez besoin

Copiez et collez les exemples donnés en ligne de commande, en remplaçant les valeurs en rouge par vos propres valeurs

N'oubliez pas que vous pouvez vérifier votre jeu de règles UFW actuel avec sudo ufw status ou sudo ufw status verbose.

## Bloquer une adresse IP

Pour bloquer toutes les connexions réseau qui proviennent d'une adresse IP spécifique, 15.15.15.51 par exemple, exécutez cette commande :

$ sudo ufw deny from 15.15.15.51

Dans cet exemple, 15.15.15.51,spécifie une adresse IP **source** de "15.15.15.51". Si vous le souhaitez, un sous-réseau, tel que 15.15.15.0/24, peut être spécifié ici à la place. L'adresse IP source peut être spécifiée dans n'importe quelle règle de pare-feu, y compris une règle **allow**.

### Bloquer les connexions à une interface réseau

Pour bloquer les connexions d'une adresse IP spécifique, par exemple 15.15.15.51, à une interface réseau spécifique, par exemple eth0, utilisez cette commande :

$ sudo ufw deny in on eth0 from 15.15.15.51

C'est la même chose que dans l'exemple précédent, avec l'ajout de in on eth01. L'interface réseau peut être spécifiée dans n'importe quelle règle de pare-feu, et c'est un excellent moyen de limiter la règle à un réseau particulier.

## Service : SSH

Si vous utilisez un serveur cloud, vous voudrez probablement autoriser les connexions SSH entrantes (port 22) afin de pouvoir vous connecter à votre serveur et le gérer. Cette section explique comment configurer votre pare-feu avec diverses règles liées à SSH.

### Autoriser les accès SSH

Pour autoriser toutes les connexions SSH entrantes, exécutez cette commande :

$ sudo ufw allow ssh

Une autre syntaxe consiste à spécifier le numéro de port du service SSH :

$ sudo ufw allow 22

### Autoriser les accès SSH entrants à partir d'une adresse IP ou d'un sous-réseau spécifique

Pour autoriser les connexions SSH entrantes à partir d'une adresse IP ou d'un sous-réseau spécifique, indiquez la source. Par exemple, si vous voulez autoriser l'ensemble du sous-réseau 15.15.15.0/24, exécutez cette commande :

$ sudo ufw allow from 15.15.15.0/24 to any port 22

### Autoriser le rsync entrant à partir d'une adresse IP ou d'un sous-réseau spécifique

Rsync, qui fonctionne sur le port 873, peut être utilisé pour transférer des fichiers d'un ordinateur à l'autre.

Pour autoriser les connexions rsync entrantes à partir d'une adresse IP ou d'un sous-réseau spécifique, il faut spécifier l'adresse IP source et le port de destination. Par exemple, si vous voulez permettre à l'ensemble du sous-réseau 15.15.15.0/24de pouvoir utiliser rsync vers votre serveur, exécutez cette commande :

$ sudo ufw allow from 15.15.15.0/24 to any port 873

## Service : Serveur web

Les serveurs web, tels qu'Apache et Nginx, écoutent généralement les requêtes sur les ports 80 et 443 pour les connexions HTTP et HTTPS, respectivement. Si votre politique par défaut pour le trafic entrant est définie sur l'abandon ou le refus, vous voudrez créer des règles qui permettront à votre serveur de répondre à ces demandes.

### Autoriser tout le trafic HTTP entrant

Pour autoriser toutes les connexions HTTP (port 80) entrantes, exécutez cette commande :

$ sudo ufw allow http

Une autre syntaxe consiste à spécifier le numéro de port du service HTTP :

$ sudo ufw allow 80

### Autoriser tous les accès HTTPS entrants

Pour autoriser toutes les connexions HTTPS (port 443) entrantes, exécutez cette commande :

$ sudo ufw allow https

Une autre syntaxe consiste à spécifier le numéro de port du service HTTP :

$ sudo ufw allow 443

### Autoriser tous les accès HTTP et HTTPS entrants

Si vous souhaitez autoriser à la fois le trafic HTTP et HTTPS, vous pouvez créer une règle unique qui autorise les deux ports. Pour autoriser toutes les connexions entrantes HTTP et HTTPS (port 443), exécutez cette commande :

$ sudo ufw allow proto tcp from any to any port 80,443

Notez que vous devez spécifier le protocole, avec proto tcp, lorsque vous spécifiez plusieurs ports.

## Service : MySQL

MySQL écoute les connexions des clients sur le port 3306. Si votre serveur de base de données MySQL est utilisé par un client sur un serveur distant, vous devez vous assurer d'autoriser ce trafic.

### Autoriser MySQL à partir d'une adresse IP ou d'un sous-réseau spécifique

Pour autoriser les connexions MySQL entrantes à partir d'une adresse IP ou d'un sous-réseau spécifique, indiquez la source. Par exemple, si vous voulez autoriser l'ensemble du sous-réseau 15.15.15.0/24, exécutez cette commande :

$ sudo ufw allow from 15.15.15.0/24 to any port 3306

### Autoriser MySQL à une interface réseau spécifique

Pour autoriser les connexions MySQL à une interface réseau spécifique - par exemple, une interface réseau privée eth1 - utilisez cette commande :

$ sudo ufw allow in on eth1 to any port 3306

## Service : PostgreSQL

PostgreSQL écoute les connexions des clients sur le port 5432. Si votre serveur de base de données PostgreSQL est utilisé par un client sur un serveur distant, vous devez vous assurer d'autoriser ce trafic.

### PostgreSQL à partir d'une adresse IP ou d'un sous-réseau spécifique

Pour autoriser les connexions PostgreSQL entrantes à partir d'une adresse IP ou d'un sous-réseau spécifique, indiquez la source. Par exemple, si vous souhaitez autoriser l'ensemble du sous-réseau 15.15.15.0/24, exécutez cette commande :

$ sudo ufw allow from 15.15.15.0/24 to any port 5432

La deuxième commande, qui permet le trafic sortant des connexions PostgreSQL établies, n'est nécessaire que si la politique OUPUT n'est pas réglée sur ACCEPT.

### Autoriser PostgreSQL à une interface réseau spécifique

Pour autoriser les connexions PostgreSQL à une interface réseau spécifique - disons que vous avez une interface réseau privée eth1, par exemple - utilisez cette commande :

$ sudo ufw allow in on eth1 to any port 5432

La deuxième commande, qui permet le trafic sortant des connexions PostgreSQL établies, n'est nécessaire que si la politique OUPUT n'est pas réglée sur ACCEPT.

## Service : Courrier électronique :

Les serveurs de messagerie, tels que Sendmail et Postfix, écoutent sur une variété de ports en fonction des protocoles utilisés pour la distribution du courrier. Si vous utilisez un serveur de courrier, déterminez quels protocoles vous utilisez et autorisez les types de trafic appropriés. Nous vous montrerons également comment créer une règle pour bloquer le courrier SMTP sortant.

### Bloquer le courrier SMTP sortant

Si votre serveur ne doit pas envoyer de courrier sortant, vous pouvez bloquer ce type de trafic. Pour bloquer le courrier SMTP sortant, qui utilise le port 25, exécutez cette commande :

$ sudo ufw deny out 25

Cela permet de configurer votre pare-feu pour qu'il **supprime** tout le trafic sortant sur le port 25. Si vous devez rejeter un service différent par son numéro de port, au lieu du port 25, il suffit de le remplacer.

### Autoriser tous les SMTP entrants

Pour permettre à votre serveur de répondre aux connexions SMTP, port 25, exécutez cette commande :

$ sudo ufw allow 25

Remarque : il est courant que les serveurs SMTP utilisent le port 587 pour le courrier sortant..

### Autoriser tous les messages IMAP entrants

Pour permettre à votre serveur de répondre aux connexions IMAP, port 143, exécutez cette commande :

$ sudo ufw allow 143

### Autoriser tous les IMAPS entrants

Pour permettre à votre serveur de répondre aux connexions IMAPS, port 993, exécutez cette commande :

$ sudo ufw allow 993

### Autoriser tous les POP3 entrants

Pour permettre à votre serveur de répondre aux connexions POP3, port 110, exécutez cette commande :

$ sudo ufw allow 110

### Autoriser tous les POP3S entrants

Pour permettre à votre serveur de répondre aux connexions POP3S, port 995, exécutez cette commande :

$ sudo ufw allow 995

## Conclusion

Cela devrait couvrir un grand nombre des commandes qui sont couramment utilisées lors de l'utilisation d'UFW pour configurer un pare-feu.