Configuration d’un compte SFTP

Table des matières

Introduction

UKG HR Service Delivery met à disposition un espace de transit pour différentes actions de masse sur les plateformes. Vous pouvez accéder à cet espace grâce à une connexion sécurisée SFTP.

Le but de ce document est de décrire les différentes étapes nécessaires à la configuration d'un compte SFTP ainsi que les moyens pour s'y connecter.

Configuration d’un compte SFTP

Etape 1 : Création d’une clé SSH

Une clé SSH (clé Secure Shell) permet de vous authentifier de manière sécurisée. La génération des clés se fait via une ligne de commande, voir Création d’une clé SSH via une ligne de commande (Secure Shell).

Etape 2 : Collecte de vos informations

Afin de vous créer un compte SFTP, nous avons besoin des informations suivantes :

- Votre ou vos adresses IP publiques (poste et/ou serveur)
- Votre clé publique

Renseignez-vous auprès de votre Service Informatique afin de connaître les adresses IP publiques qui seront utilisées.

Etape 3 : Communication de vos identifiants

Une fois autorisé sur notre serveur, UKG HR Service Delivery vous communique les informations nécessaires pour vous connecter à votre compte :

- Votre identifiant
- L’adresse IP du serveur SFTP
- Le port du serveur

Renseignez-vous auprès de votre Service Informatique afin de vous assurer que vous pouvez accéder au serveur à l’adresse IP et sur le port demandé.

Etape 4 : Configuration de votre client SFTP

Une fois toutes les informations transmises, vous pouvez configurer votre client afin de vous connecter.

- Exemple de configuration avec FileZilla : Configuration d’un client SFTP (FileZilla)
- Exemple de script d’automatisation en Python : Exemple de script d’automatisation

Vous trouverez toutes les informations de connexions aux serveurs SFTP de UKG HR Service Delivery dans la documentation disponible à cette adresse : https://doc.people-doc.com/client/guides/synchronization/.

Principes de l’authentification par clé

L’authentification par clé fonctionne grâce à 3 composants :

- Une clé privée : Elle permet de prouver son identité au serveur SFTP.
  La clé privée ne doit pas nous être envoyée. Elle devra être conservée et installée sur vos postes/serveurs nécessitant de se connecter à notre serveur.

- Une passphrase (optionnelle) : Elle permet de sécuriser la clé privée.

- Une clé publique : Elle permet au serveur d’autoriser la clé privée correspondante.
  La **clé publique (se terminant par “.pub”)** est à fournir à UKG HR Service Delivery afin de configurer votre accès. **UKG HR Service Delivery n'accepte que les clés SSH publiques au format OpenSSH.**

Pour plus d’informations sur la sécurité liée à l’utilisation de clés SSH:
https://www.ssh.com/ssh/openssh/#sec-What-Risks-Are-Associated-with-SSH-Keys

Création d'une clé SSH via une ligne de commande (Secure Shell)

L'outil **ssh-keygen** existe sous Windows, Linux et Mac, la syntaxe de génération d'une clé est donc commune.

Dans un terminal ou invite de commande Windows, vous pouvez utiliser l'assistant en tapant uniquement :

- *ssh-keygen*

Ou utiliser la syntaxe suivante en modifiant les valeurs selon le tableau ci-dessous :

- *ssh-keygen -t **KEY_TYPE** -b **BITS** -C "**COMMENT**" -f "**LOCATION_&_NAME**"*

Nous vous conseillons d'utiliser la syntaxe complète afin de pouvoir spécifier explicitement les caractéristiques de votre clé (voir les recommandations du tableau ci-dessous).

En complément, il est recommandé d'attribuer une passphrase à la génération de votre clé SSH afin d'en assurer la sécurité. Cette passphrase ne devra être connue que de vous seul.
Les deux commandes citées précédemment vous inviterons à la définir.

Attribuez un commentaire à chacune de vos clés SSH afin de les différencier, ce qui permet de communiquer plus facilement en cas d'autorisations de plusieurs clés publiques.

Section	Caractéristiques de Clé
KEY_TYPE	Voir notre documentation en ligne pour les types et format de clé supportés et recommandés, ainsi que les tailles associées.
COMMENT	• Identification du propriétaire de la clé (adresse email) *Exemple : martin_dupont@entreprise.fr*
LOCATION_&_NAME	Emplacement de création des clés • Windows : *c:\users\Martin_Dupont\id_rsa_martin_dupont* • Linux / Mac : */home/martin_dupont/.ssh/id_rsa_martin_dupont*

 # Exemples via syntaxe complète

Clé SSH ed25519

ssh-keygen -t ed25519 -C “email” -f /path/id_ed25519_owner

```
your_user@laptop-your-user:~$ ssh-keygen -t ed25519 -C your_user@enterprise -f /home/your_user/.ssh/id_ed25519_your_user
Generating public/private ed25519 key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/your_user/.ssh/id_ed25519_your_user.
Your public key has been saved in /home/your_user/.ssh/id_ed25519_your_user.pub.
The key fingerprint is:
SHA256:buneMTLhvevcFSP/ucDBsjCczZrZbOtBNFCD7Xu+/mQ your_user@enterprise
The key's randomart image is:
+--[ED25519 256]--+
|          .+o    |
|          ....   |
|           .o    |
|         . =.o   |
|        S= =ooo  |
|       o +X.++.o |
|        B++*oooE |
|       o =.=oo+..|
|       .o.B+ooo+o|
+----[SHA256]-----+
```

Clé SSH rsa-sha2-512

ssh-keygen -b 4096 -t rsa-sha2-512 -C “email” -f /path/id_rsa-sha2-512_owner

Clé SSH rsa-sha2-256

ssh-keygen -b 4096 -t rsa-sha2-256 -C “email” -f /path/id_rsa_sha2-256_owner

Clé SSH rsa

ssh-keygen -b 4096 -t rsa -C “email” -f /path/id_rsa_owner

 # Exemple via l'assistant

ssh-keygen

```
your_user@laptop-your-user:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/your_user/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/your_user/.ssh/id_rsa.
Your public key has been saved in /home/your_user/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:RwJN2iggKWre2VhttZCcKL5/GJz8B4YaTO/mfn6pehc your_user@laptop-your-user
The key's randomart image is:
+---[RSA 2048]----+
|...   +o+        |
|o. o . X..       |
|o . o + = o      |
|.. o o o +       |
|o + X + S .      |
| . B O o E       |
|    = = . o      |
|   . = = =       |
|    ++*o=        |
+----[SHA256]-----+
```

Configuration d’un client SFTP (FileZilla)

Téléchargez et installez FileZilla : https://filezilla-project.org/download.php?type=client

Cliquez sur Fichier -> Gestionnaire de Sites -> Nouveau Site.

Remplissez les champs selon le serveur sur lequel vous souhaitez vous connecter.

Protocole	SFTP
Hôte	Adresse (DNS) du serveur SFTP *exemple : adresse_sftp_server.ukg.com*
Port	9030
Type d’authentification	Fichier de clé
Identifiant	Votre login transmis par UKG HR Service Delivery
Fichier de clé	Chemin vers votre clé privée *exemple : c:\users\Martin_Dupont\id_rsa_Martin_Dupont*

Enfin, cliquez sur Connexion (le profil du Site sera automatiquement enregistré).

Si une passphrase a été définie lors de la création de votre paire de clés (publique et privée), FileZilla vous invitera à la renseigner au moment de la connexion.

Lors de la première connexion, un avertissement s’affiche permettant de vous assurer que vous vous connectez bien au serveur souhaité.

Si l'empreinte est identique à celle du serveur auquel vous souhaitez accéder [RSA key fingerprint ou SSH public key (PEM format) du serveur sur la documentation officielle], vous pouvez approuver ce serveur et l’associer à la connexion.

Dans le cas contraire, merci de nous en faire part.

Exemple de script d'automatisation

Pour la production, il est vivement recommandé d'automatiser le transfert des fichiers à partir de vos serveurs vers notre serveur SFTP.

Exemple de script Python utilisant le module paramiko.

```
#!/usr/bin/env python

import paramiko
paramiko.util.log_to_file('/tmp/paramiko.log')

# Connection

host = 'adresse_sftp_server.ukg.com'
port = 9030
transport = paramiko.Transport((host, port))

# Authentication

username = 'xxxx'
key_path = '/home/xxx/.ssh/id_rsa'
key_pass = ''

my_key = paramiko.RSAKey.from_private_key_file(key_path, key_pass)
transport.connect(username=username, pkey=Mikey)

# SFTP client

sftp = paramiko.SFTPClient.from_transport(transport)

# Upload using .filepart extension to prevent remote processing during the transfer

local_path = '/home/xxxx/ndmat_yyyy_zzzz_sal_20150909.csv'
remote_path = 'in/sal/ndmat_yyyy_zzzz_sal_20150909.csv'
sftp.put(local_path, remote_path + '.filepart')
sftp.rename(remote_path + '.filepart', remote_path)

# Stop gracefully

sftp.close()
transport.close()
```