

RaspFullSetup

Net and Light

2023 - 2024

Résumé

Cette documentation décrit le script Python "RaspFullSetup.py" conçu pour se connecter à plusieurs Raspberry Pi via SSH et effectuer diverses opérations, telles que le transfert de fichiers et l'exécution de commandes, en utilisant les bibliothèques `paramiko`, `requests`, et `mysql.connector`. Le script lit également la configuration à partir d'un fichier JSON.

Table des matières

1	Introduction	2
2	Configuration	2
2.1	Importations	2
2.2	Configuration	2
3	Connexion à la Base de Données	2
4	Récupération des Adresses IP	3
5	Envoi de Requêtes HTTP	3
6	Connexion SSH et Exécution de Commandes	3
7	Boucle Principale	5

1 Introduction

Ce script Python utilise `paramiko` pour établir des connexions SSH et effectuer des opérations SFTP sur des Raspberry Pi. Il lit également la configuration à partir d'un fichier JSON et utilise `requests` pour envoyer des requêtes HTTP.

2 Configuration

2.1 Importations

```
1 import paramiko # Import paramiko for SSH and SFTP
    connectivity and operations.
2 import os
3 import requests # Imports the requests module to make HTTP
    requests
4 import json # Imports the json module for parsing and
    generating JSON data
5 from ip_addresses import fetch_latest_ip_addresses #
    Imports the fetch_latest_ip_addresses function from
    ip_addresses module
6 from mysql.connector import connect # Imports the connect
    function from mysql.connector module for database
    connections
```

Listing 1 – Importations

2.2 Configuration

La configuration est chargée à partir d'un fichier JSON nommé `configHermes.json`, qui contient les informations nécessaires pour se connecter aux Raspberry Pi, ainsi que des chemins de fichiers locaux et distants.

```
1 with open('configHermes.json', 'r') as config_file: # Opens
    the file in read mode
2     config = json.load(config_file) # Loads the JSON content
    and converts it into a Python dictionary
```

Listing 2 – Chargement de la configuration

3 Connexion à la Base de Données

Le script utilise `mysql.connector` pour se connecter à une base de données MySQL en utilisant les informations de connexion fournies dans le fichier de configuration.

```
1 # Define a connection object
2 conn = connect(
3     user=config['Login_Turtle']['user'],
4     password=config['Login_Turtle']['password'],
```

```
5     host=config['Login_Turtle']['host'],
6     database=config['Login_Turtle']['database'])
```

Listing 3 – Connexion à la base de données

4 Récupération des Adresses IP

Le script utilise la fonction importée `fetch_latest_ip_addresses()` pour récupérer les adresses IP des Raspberry Pi.

```
1 ip_addresses = fetch_latest_ip_addresses()
```

Listing 4 – Récupération des adresses IP

5 Envoi de Requêtes HTTP

Le script envoie des requêtes HTTP pour arrêter des scripts en cours d'exécution sur les Raspberry Pi.

```
1 for ip_addresses in ip_addresses.values():
2     try:
3         requests.get(f"http://{ip_addresses['RASP_catch
4             ']}:8000/kill")
5     except Exception as e:
6         print(e)
```

Listing 5 – Envoi de requêtes HTTP pour arrêter les scripts

6 Connexion SSH et Exécution de Commandes

La fonction `ssh_and_run()` se connecte à un Raspberry Pi via SSH, transfère des fichiers et exécute plusieurs commandes pour configurer l'environnement et lancer un script.

```
1 def ssh_and_run(ip, username, password, local_file,
2     target_dest, command):
3     try:
4         client = paramiko.SSHClient()
5         client.set_missing_host_key_policy(paramiko.
6             AutoAddPolicy())
7         client.connect(ip, username=username, password=
8             password)
9
10        try:
11            sftp = client.open_sftp()
12            sftp.put(local_file, target_dest)
13            sftp.close()
14            print("Fichiers Transférés avec succès ! ")
```

```

12     except Exception as e:
13         print(f"Le transfert a rencontré un problème : {e}")
14
15     print("Creating a virtual environment on the
16           Raspberry Pi")
17     stdin, stdout, stderr = client.exec_command("python3
18           -m venv ~/venv")
19     output = stdout.read().decode(errors='ignore')
20     error = stderr.read().decode(errors='ignore')
21     print(f"Terminal de {ip}: {output}")
22     if error:
23         print(f"Erreur de {ip}: {error}")
24
25     print("Activating the virtual environment")
26     stdin, stdout, stderr = client.exec_command("source
27           ~/venv/bin/activate")
28     output = stdout.read().decode(errors='ignore')
29     error = stderr.read().decode(errors='ignore')
30     print(f"Terminal de {ip}: {output}")
31     if error:
32         print(f"Erreur de {ip}: {error}")
33
34     print("Installing asyncua in the virtual environment")
35     )
36     stdin, stdout, stderr = client.exec_command("~/venv/
37           bin/pip install asyncua")
38     output = stdout.read().decode(errors='ignore')
39     error = stderr.read().decode(errors='ignore')
40     print(f"Terminal de {ip}: {output}")
41     if error:
42         print(f"Erreur de {ip}: {error}")
43
44     print("Running the script in the virtual environment")
45     )
46     stdin, stdout, stderr = client.exec_command("nohup ~/
47           venv/bin/python " + command + " > /dev/null 2>&1 &
48           disown")
49     exit_status = stdout.channel.recv_exit_status() #
50         Get the exit status of the command
51     print(f"Terminal de {ip}: {output}")
52     if error:
53         print(f"Erreur de {ip}: {error}")
54
55     print(f"Script exécuté avec succès sur {ip}")
56     client.close()
57
58 except Exception as e:

```

```
50 print(f"Echec de l'exécution du script sur {ip}: {e}")
    )
```

Listing 6 – Fonction `ssh_and_run()`

7 Boucle Principale

Le script parcourt les adresses IP des Raspberry Pi et appelle la fonction `ssh_and_run()` pour chaque adresse.

```
1 for ip_addresses in ip_addresses.values():
2     ip = ip_addresses['RASP_catch']
3     ssh_and_run(ip, username, password, local_file_1,
        target_dest_1, Command)
```

Listing 7 – Boucle principale