# RaspFullSetup

Net and Light

2023 - 2024

**Résumé**

Cette documentation décrit un script Python 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

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

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.

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

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

Listing 1 – 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.

```
# Define a connection object
conn = connect(
      user=config['Login_Turtle']['user'],
      password=config['Login_Turtle']['password'],
      host=config['Login_Turtle']['host'],
      database=config['Login_Turtle']['database'])
```

Listing 2 – Connexion à la base de données

# 4 Récupération des Adresses IP

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

```
ip_addresses = fetch_latest_ip_addresses()
```

Listing 3 – 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.

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

Listing 4 – 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.

```
def ssh_and_run(ip, username, password, local_file,
   target_dest, command):
    try:
        client = paramiko.SSHClient()
        client.set_missing_host_key_policy(paramiko.
           AutoAddPolicy())
        client.connect(ip, username=username, password=
           password)

        try:
            sftp = client.open_sftp()
            sftp.put(local_file, target_dest)
            sftp.close()
            print("Fichiers Transférés avec succès ! ")
        except Exception as e:
            print(f"Le transfert a rencontré un problème : {e
               }")

        print("Creating a virtual environment on the
           Raspberry Pi")
```

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

Listing 5 – 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.

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

Listing 6 – Boucle principale