

# RaspQuickLaunch

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

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

## 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.

```
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
7
8 with open('configHermes.json', 'r') as config_file: # Opens
   the file in read mode
9     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.

```
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 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.

```
1 ip_addresses = fetch_latest_ip_addresses()
2 print(f"IP addresses: {ip_addresses.values()}")
```

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.

```
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 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.

```
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 ! ")
15            sftp = client.open_sftp()
16            sftp.put(local_file_2, target_dest_2)
17            sftp.close()
18            print("Fichiers Transférés avec succès ! ")
19            sftp = client.open_sftp()
```

```
17         sftp.put(local_file_3, target_dest_3)
18         sftp.close()
19         print("Fichiers Transférés avec succès ! ")
20         sftp = client.open_sftp()
21         sftp.put(local_file_4, target_dest_4)
22         sftp.close()
23         print("Fichiers Transférés avec succès ! ")
24         sftp = client.open_sftp()
25         sftp.put(local_file_5, target_dest_5)
26         sftp.close()
27         print("Fichiers Transférés avec succès ! ")
28     except Exception as e:
29         print(f"Le transfert a rencontré un problème : {e}
30             ")
31
32     print("Running the script in the virtual environment"
33         )
34     stdin, stdout, stderr = client.exec_command("nohup ~/
35         venv/bin/python " + command + " > /dev/null 2>&1 &
36         disown")
37     exit_status = stdout.channel.recv_exit_status() #
38         Get the exit status of the command
39     client.close()
40     print(f"Exit status of the command on {ip}: {
41         exit_status}")
42
43 except Exception as e:
44     print(f"Erreur de {ip}: {e}")
45
46 print(f"Script exécuté avec succès sur {ip}")
```

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.

```
1 for ip_adresses in ip_adresses.values():
2     print(f"Trying to connect to {ip_adresses['RASP_catch']}")
3     ssh_and_run(ip_adresses['RASP_catch'], username, password
4         , local_file_1, target_dest_1, Command)
5     print(f"Script exécuté avec succès sur {ip_adresses}")
```

Listing 6 – Boucle principale