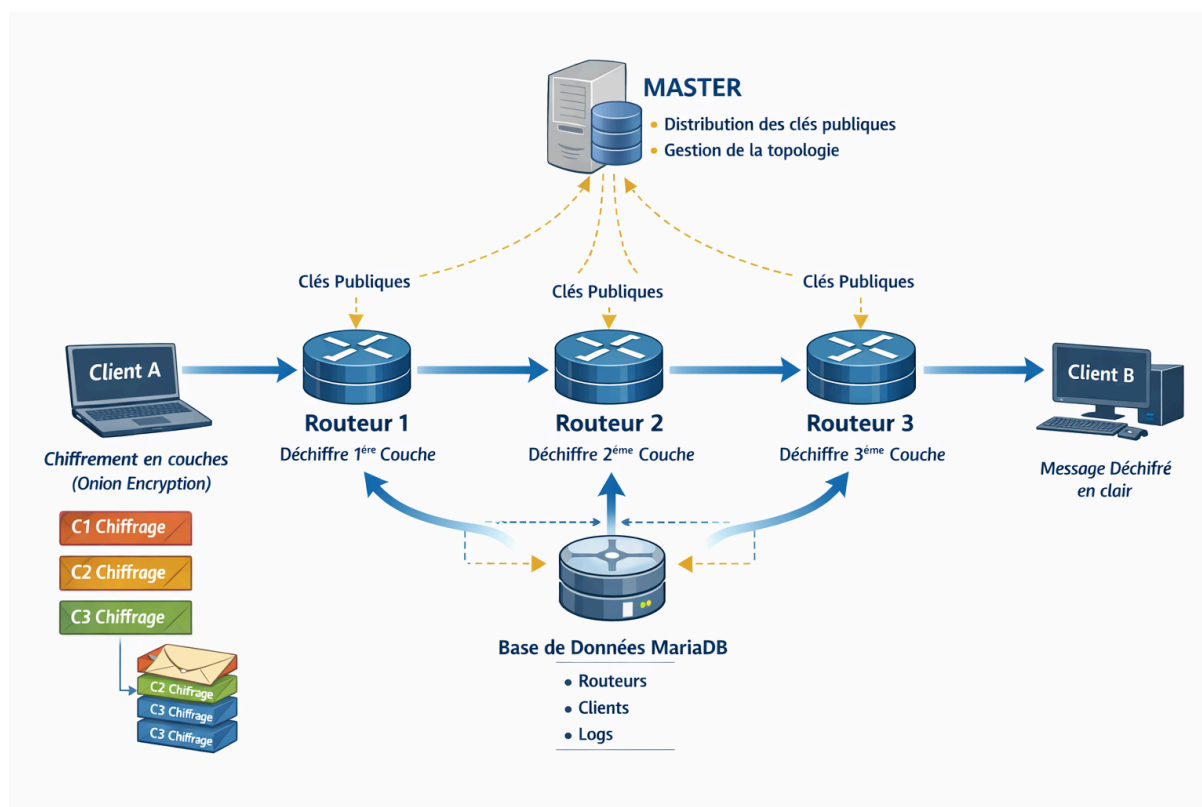


Guide d'Installation et de Configuration du Système de Routage en Oignon



Introduction	4
1. Installation des Machines Virtuelles	4
Installation de VirtualBox :	4
1 Création des VMs :	5
1. 1 Configuration du Réseau :	7
2. Installation de MariaDB	8
2.1 Installation de MariaDB sur une MV Debian :	8
2.2 Connexion et Création de la Base de Données et des Tables :	9
2.2.1 Création d'un Utilisateur Dédié :	9
2.2.2 Création de la Base de Données et des Tables :	9
3. Installation des Scripts Python	12
3.1. Installation des Dépendances :	12
3.2. Mise en Place des Scripts :	12
4. Lancement / Configuration des scripts :	14
4.1. Lancer le Serveur Master (Master Server) :	14
Vue d'ensemble des connexions :	16
Logs d'activité :	18
4.2. Lancer un Routeur	19
4.3. Lancer un Client	19
1. Envoi de Messages :	20
2. Réception de Messages :	21
3. Gestion des Routeurs :	22

Introduction

Ce projet consiste à créer un système de communication anonyme entre plusieurs utilisateurs, en utilisant le "routage en oignon". Chaque message est chiffré plusieurs fois, chaque routeur ne connaissant que son voisin direct, garantissant ainsi l'anonymat des échanges.

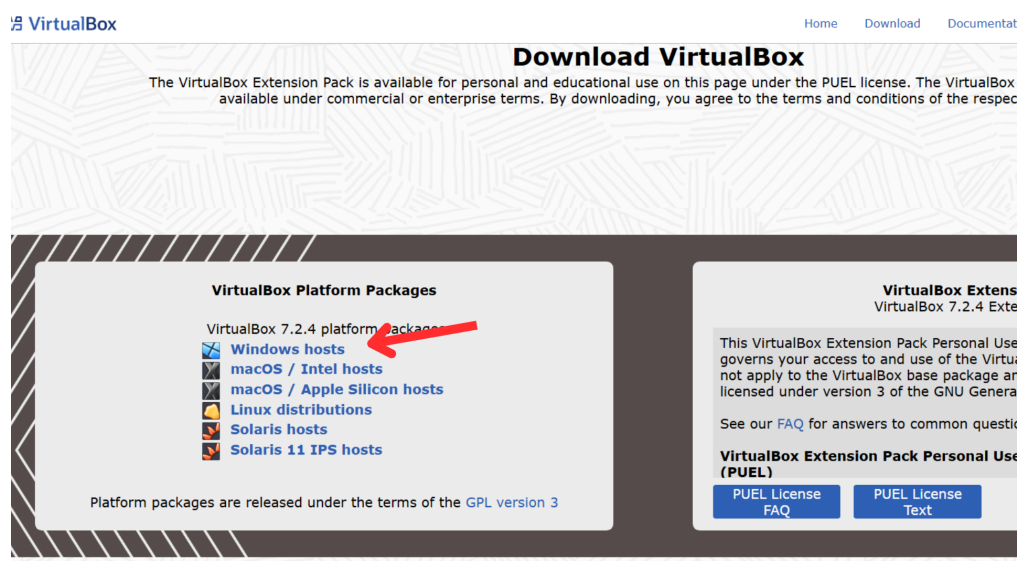
Le but est de permettre à des clients d'échanger des messages de manière sécurisée via des routeurs virtuels et un serveur maître qui gère la topologie du réseau. Ce guide vous expliquera étape par étape comment installer et configurer le système.

1. Installation des Machines Virtuelles

Pour ce projet, nous utiliserons Oracle VirtualBox pour créer des machines virtuelles qui hébergeront nos routeurs et clients. Vous devrez installer deux systèmes d'exploitation : **Debian** et **Windows**.

Installation de VirtualBox :

Téléchargez et installez [Oracle VirtualBox](#)



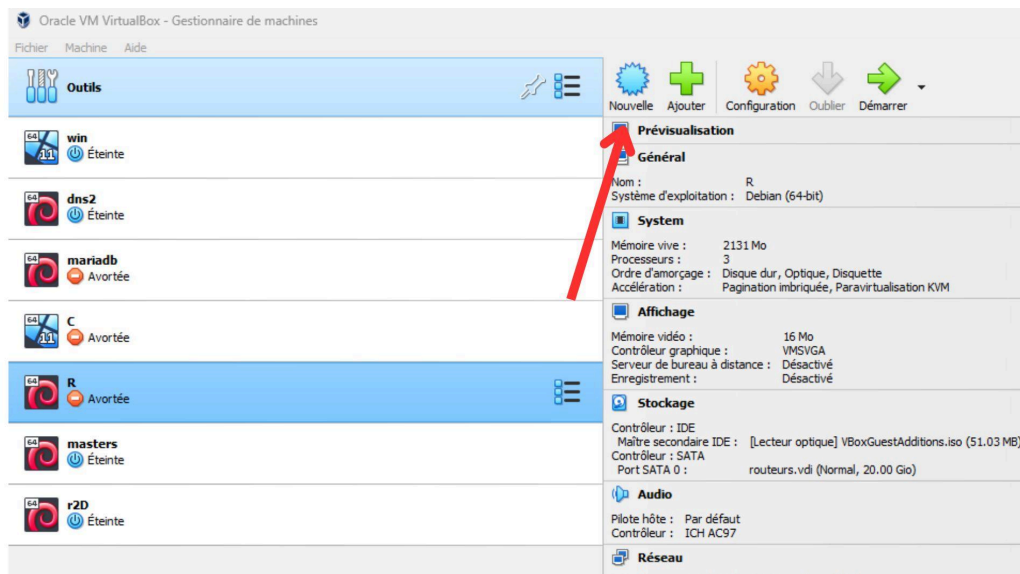
1. Création des VMs :

1 - Créez deux VM Debian et une VM Windows :

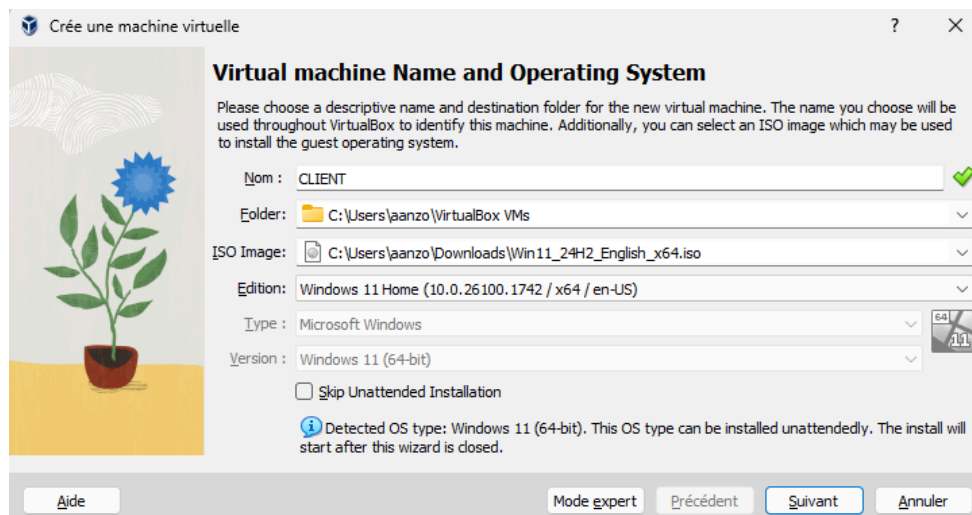
Debian 13 : Téléchargez l'ISO depuis [Debian.org](#).

Windows 11 : Téléchargez l'ISO depuis [Microsoft.com](#)

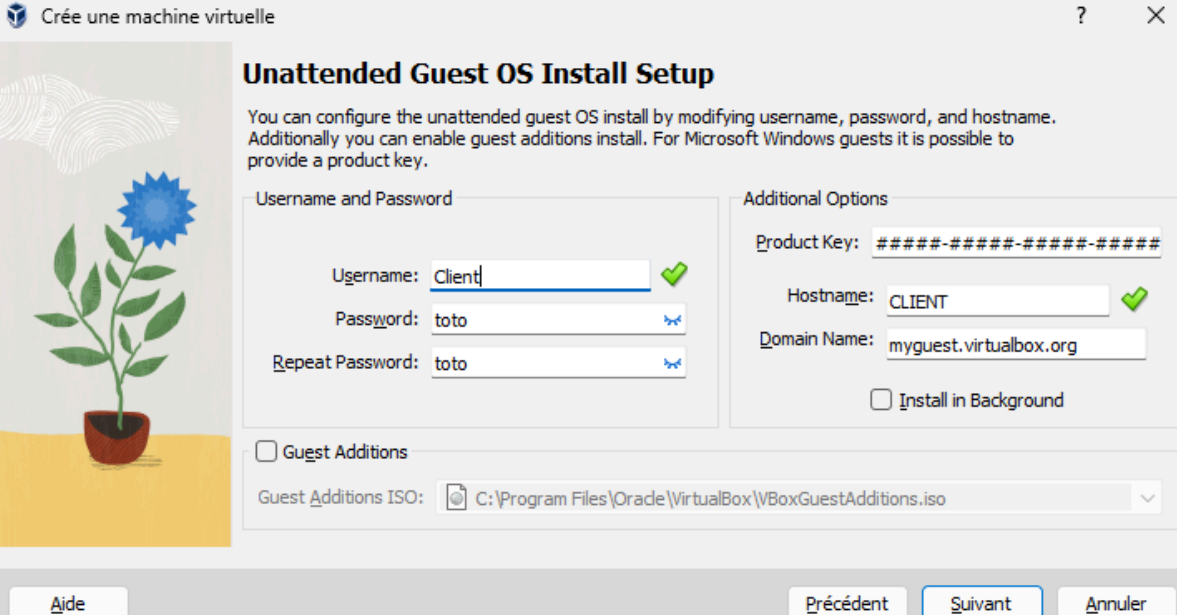
2 - Suivez les étapes d'installation de chaque système d'exploitation



Sélectionner l'iso que vous avez précédemment téléchargé windows ou debian



n'oubliez pas de modifier votre mot de passe !



Crée une machine virtuelle

Unattended Guest OS Install Setup

You can configure the unattended guest OS install by modifying username, password, and hostname. Additionally you can enable guest additions install. For Microsoft Windows guests it is possible to provide a product key.

Username and Password

Username: ✓

Password: ✕

Repeat Password: ✕

Additional Options

Product Key:

Hostname: ✓

Domain Name:

☐ Install in Background

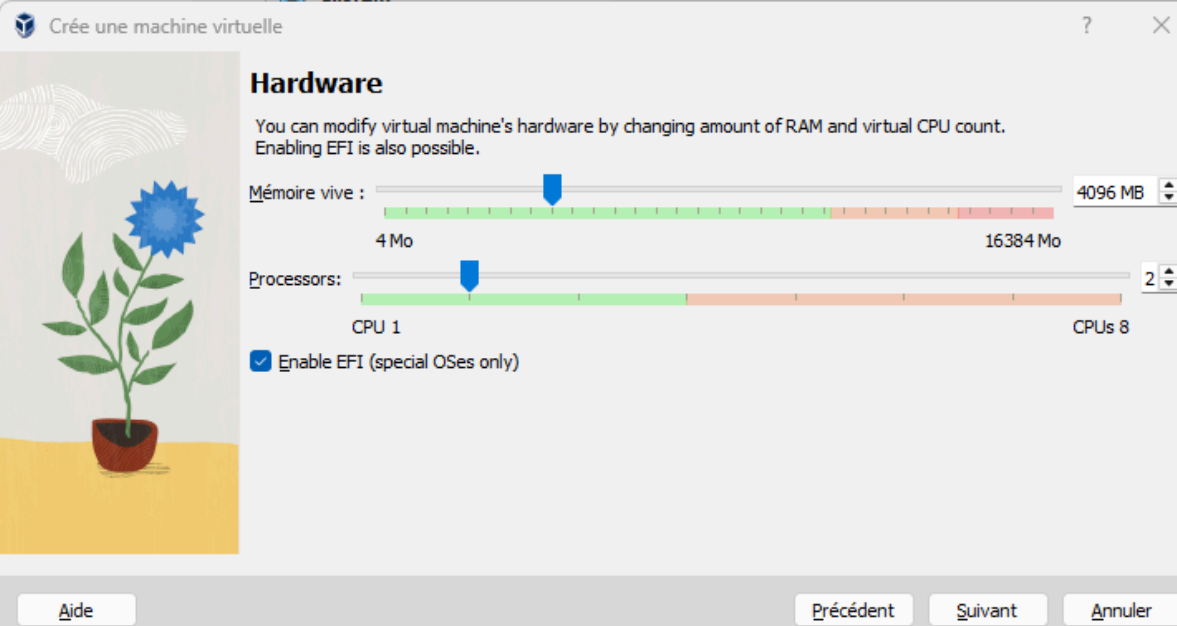
☐ Guest Additions

Guest Additions ISO:

Aide Précédent Suivant Annuler

attribuez de la RAM ainsi que des processeurs

RAM Minimum 2 Go pour Votre MV Linux et 5Go Pour Windows



Crée une machine virtuelle

Hardware

You can modify virtual machine's hardware by changing amount of RAM and virtual CPU count. Enabling EFI is also possible.

Mémoire vive : 4096 MB

4 Mo 16384 Mo

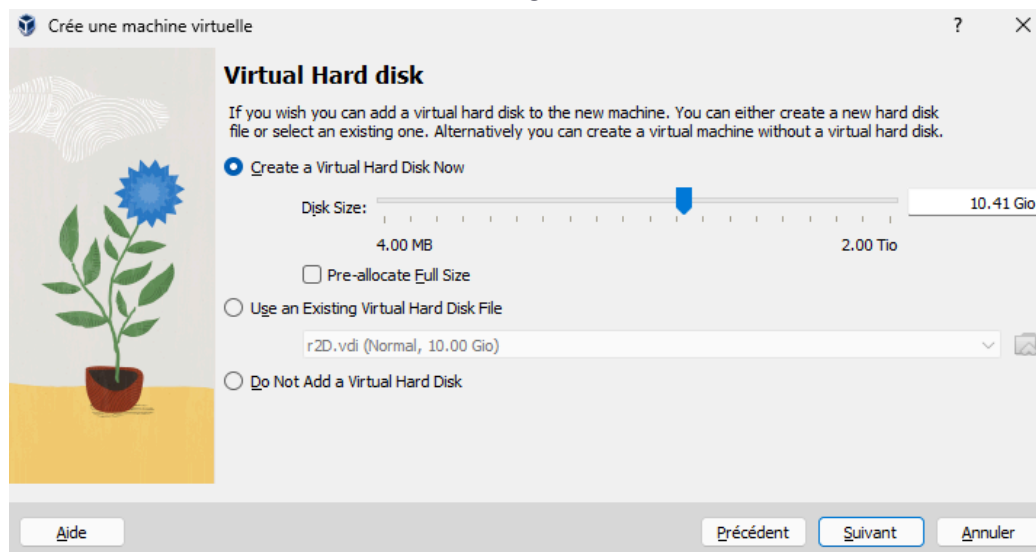
Processors: 2

CPU 1 CPUs 8

☒ Enable EFI (special OSes only)

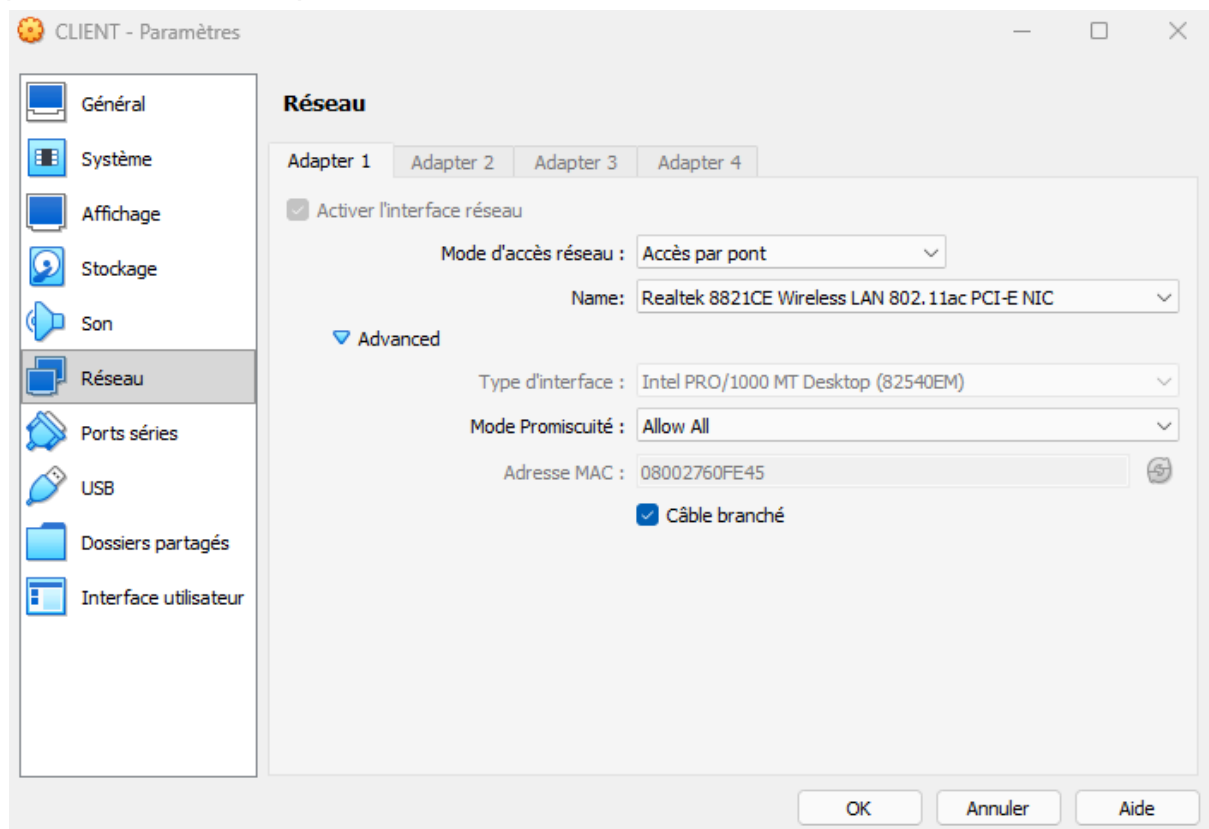
Aide Précédent Suivant Annuler

ensuite, vous allouez votre stockage virtuel 11Go



1. 1 Configuration du Réseau :

Configurez le réseau des VMs en mode **Accès par pont** pour qu'elles puissent communiquer entre elles.



Vérifiez l'attribution des adresses IP avec `ip a` (Debian) ou `ipconfig` (Windows).

2. Installation de MariaDB

MariaDB est la base de données que nous utiliserons pour stocker les informations de routage et de clés de chiffrement. Voici comment l'installer et la configurer.

2.1 Installation de MariaDB sur une MV Debian :

Ouvrez un terminal et mettez à jour votre système :

```
sudo apt update
```

puis installez mariaDB

```
sudo apt install mariadb-server
```

Démarrez MariaDB :

```
sudo systemctl start mariadb
```

Vérifiez que MariaDB fonctionne correctement :

```
mariadb --version
```

```
sudo systemctl status mariadb
```

résultat attendu :

```
root@r2D:/home/toto# sudo systemctl status mariadb
● mariadb.service - MariaDB 10.11.14 database server
   Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; preset: enabled)
   Active: active (running) since Tue 2025-12-30 16:19:06 CET; 6min ago
     Docs: man:mariadb(8)
           https://mariadb.com/kb/en/library/systemd/
  Process: 582 ExecStartPre=/usr/bin/install -m 755 -o mysql -g root -d /var/run/mysqld (co
 Process: 586 ExecStartPre=/bin/ch -c systemctl unset-environment WSRFP START POSITION /c
```


2.2 Connexion et Création de la Base de Données et des Tables :

2.2.1 Création d'un Utilisateur Dédié :

1 - Connectez-vous à MariaDB en tant que **root**

```
mysql -u root -p
```

2 - Créez un utilisateur **toto** et définissez son mot de passe :

```
CREATE USER 'toto'@'%' IDENTIFIED BY 'toto';
```

Si vous choisissez de renommer la BDD ou votre utilisateur, cela entraînera une modification du code dans la fonction `get_db_connection` du code `serveur_master`.

```
def get_db_connection(db_ip):  
    """Retourne une connexion a la base de donnees."""  
    try:  
        conn = mariadb.connect(  
            host=db_ip,  
            user="toto",  
            password="toto",  
            database="table_routage"  
        )
```

2.2.2 Création de la Base de Données et des Tables :

1 - Créez la base de données :

```
CREATE DATABASE table_routage;
```

2 - Créez les tables nécessaires pour stocker les logs, les routeurs et les clients

```
USE table_routage;
```

```
CREATE TABLE logs (  
    id INT(11) NOT NULL AUTO_INCREMENT,  
    message_id VARCHAR(50) DEFAULT NULL,  
    routeur VARCHAR(50) DEFAULT NULL,  
    timestamp DATETIME DEFAULT CURRENT_TIMESTAMP,  
    PRIMARY KEY (id)  
);
```

```
CREATE TABLE routeurs (  
    id INT(11) NOT NULL AUTO_INCREMENT,  
    nom VARCHAR(50) DEFAULT NULL,  
    adresse_ip VARCHAR(50) DEFAULT NULL,  
    type VARCHAR(50) DEFAULT NULL,  
    next_hop VARCHAR(50) DEFAULT NULL,  
    port INT(11) DEFAULT NULL,  
    cle_publique VARCHAR(500) DEFAULT NULL,  
    PRIMARY KEY (id)  
);
```

```
CREATE TABLE clients (  
    id INT(11) NOT NULL AUTO_INCREMENT,  
    nom VARCHAR(100) DEFAULT NULL,  
    adresse_ip VARCHAR(50) DEFAULT NULL,  
    port INT(11) DEFAULT NULL,  
    cle_publique TEXT DEFAULT NULL,  
    next_hop VARCHAR(50) DEFAULT NULL,  
    PRIMARY KEY (id)  
);
```

3 - Accordez à l'utilisateur `toto` les privilèges nécessaires pour accéder à la base de données `table_routage` depuis n'importe quelle adresse IP :

```
GRANT ALL PRIVILEGES ON table_routage.* TO 'toto'@'%';
```

4 - Appliquez les privilèges :

```
FLUSH PRIVILEGES;
```

5 - Quittez MariaDB :

```
EXIT;
```

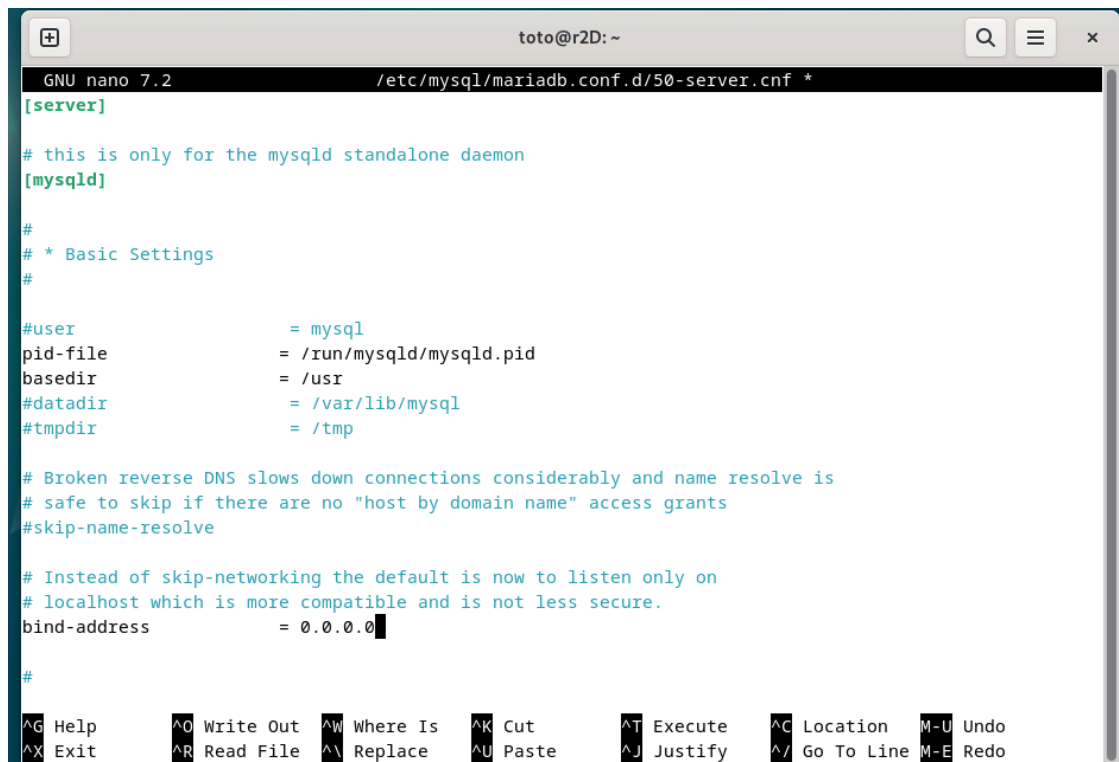
6 - Autoriser l'accès externe à MariaDB

ouvrez le fichier de configuration de MariaDB :

```
sudo nano /etc/mysql/mariadb.conf.d/50-server.cnf
```

Modifiez la ligne `bind-address` pour :

```
bind-address = 0.0.0.0
```



```
toto@r2D: ~
GNU nano 7.2 /etc/mysql/mariadb.conf.d/50-server.cnf *
[server]

# this is only for the mysqld standalone daemon
[mysqld]

#
# * Basic Settings
#

#user                = mysql
pid-file             = /run/mysqld/mysqld.pid
basedir              = /usr
#datadir             = /var/lib/mysql
#tmpdir              = /tmp

# Broken reverse DNS slows down connections considerably and name resolve is
# safe to skip if there are no "host by domain name" access grants
#skip-name-resolve

# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address         = 0.0.0.0

#

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location  M-U Undo
^X Exit      ^R Read File ^N Replace   ^U Paste     ^J Justify   ^_ Go To Line M-E Redo
```

Redémarrez MariaDB :

`sudo systemctl restart mariadb`

⚠ **Notez bien l'IP de votre MV mariadb, relevez-la avec la commande `ip a`**

3. Installation des Scripts Python

3.1. Installation des Dépendances :

Avant de pouvoir exécuter les scripts Python, vous devez installer certaines dépendances qui sont utilisées dans le projet. Assurez-vous d'avoir **Python 3** installé sur votre machine.

Lien de téléchargement : [Download Python | Python.org](https://www.python.org/downloads/)

Vous pouvez vérifier la version de Python avec la commande suivante :

```
python3 --version
```

Ensuite, vous devez installer les bibliothèques nécessaires. Les deux principales dépendances sont **PyQt6** pour l'interface graphique et **mariadb** pour la gestion de la base de données. Voici comment installer ces dépendances :

1 - Installez PyQt6 pour l'interface graphique :

```
pip install PyQt6
```

2 - Installez MariaDB pour la connexion à la base de données :

```
pip install mariadb
```

3.2. Mise en Place des Scripts :

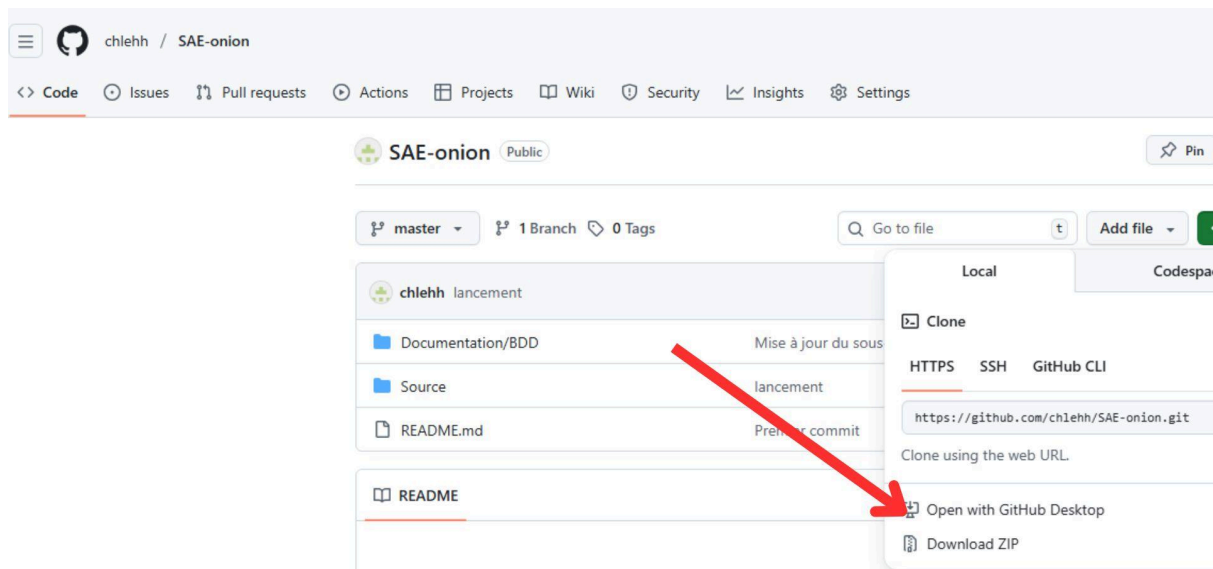
Une fois les dépendances installées, vous pouvez maintenant configurer et mettre en place les scripts Python nécessaires au fonctionnement du système.

Étape 1 : Téléchargez et placez les scripts

Télécharger le Projet sur Windows :

Accédez à ce lien GitHub et téléchargez l'ensemble des codes :

<https://github.com/chlehh/SAE-onion>



Télécharger le Projet sur Debian :

Installer Git :

Si Git n'est pas encore installé, ouvrez un terminal et installez-le avec la commande suivante :

```
sudo apt update  
sudo apt install git
```

Cloner le projet depuis GitHub :

```
git clone https://github.com/chlehh/SAE-onion
```

Accéder au répertoire du projet :

```
cd SAE-onion
```

Créer un Environnement Virtuel :

Sur Debian Avant d'installer les dépendances, créez un environnement virtuel pour isoler les bibliothèques Python du système. Exécutez cette commande dans le répertoire du projet :

```
python3 -m venv venv
```

Puis activer le :

```
source venv/bin/activate
```

Installation des dépendances :

Pour installer les dépendances, il vous suffit de consulter le fichier **requirements.txt**

Il vous suffit de lancer la commande suivante :

```
pip install -r requirements.txt
```

4. Lancement / Configuration des scripts :

Une fois les dépendances installées et l'environnement virtuel configuré, vous êtes prêt à lancer les scripts nécessaires pour faire fonctionner le système de routage en oignon.

4.1. Lancer le Serveur Master (Master Server) :

Étapes pour lancer le serveur Master :

Accédez au répertoire Master

chemin : SAE-onion — > Source —> Master

dans un terminal, lancer la commande suivante :

```
python main.py <IP_BDD> <PORT_MASTER>
```

Remplacez **<IP_BDD>** par l'adresse IP de votre base de données MariaDB relevez Précédement .

Remplacez **<PORT_MASTER>** par le port sur lequel le serveur Master écoutera les connexions des clients et des routeurs

Cela lancera l'interface graphique du serveur Master, où vous pourrez voir les informations sur les routeurs et les clients connectés, ainsi que les logs.

Serveur Master - Port 5000

—

📄

✕

Serveur Master - Routage en Oignon

DB: 192.168.1.194 | Port: 5000

Statistiques

Routeurs: 0

Clients: 0

Logs: 0

Routeurs

Clients

Logs

Routeurs Actifs

Nom	IP	Port	Cle Publique	Actions
-----	----	------	--------------	---------

Rafraichir

Statut

[03:50:25] Interface Master initialisee

[03:50:25] En attente de connexions sur le port 5000

Routeurs

Clients

Logs

Clients Connectes

Nom	IP	Port	Actions
-----	----	------	---------

Rafraichir

Serveur Master - Routage en Oignon

DB: 192.168.1.194 | Port: 5000

Statistiques

Routeurs: 0

Clients: 0

Logs: 0

Routeurs Clients Logs

Logs des Messages

ID	Message ID	Routeur	Timestamp

Rafraichir

Effacer les logs

Vue d'ensemble des connexions :

- **Liste des clients** : L'interface affiche une liste des **clients** actuellement connectés au serveur. Vous pouvez voir des informations comme :
 - Le **nom** du client.
 - L'**adresse IP** et le **port** du client.
 - Le **statut** de la connexion (connecté, déconnecté, en attente, etc.).

Serveur Master - Port 5000

Serveur Master - Routage en Oignon

DB: 192.168.1.194 | Port: 5000

Statistiques

Routeurs: 2 Clients: 1 Logs: 0

Routeurs Clients Logs

Clients Connectes

	Nom	IP	Port	Actions
1	amine	192.168.1.194	5034	Supprimer

Rafraichir

- **Liste des routeurs** : De même, l'interface affiche une liste des **routeurs** connectés au serveur. Les informations suivantes sont affichées :
 - Le **nom** du routeur.
 - L'**adresse IP** et le **port** du routeur.
 - La **clé publique** du routeur .

Statistiques

Routeurs: 2 Clients: 1 Logs: 0

Routeurs Clients Logs

Routeurs Actifs

	Nom	IP	Port	Cle Publique	Action
1	R1	192.168.1.194	6666	65537,121103	Supprimer
2	R2	192.168.1.194	6667	5,48929	Supprimer

Rafraichir

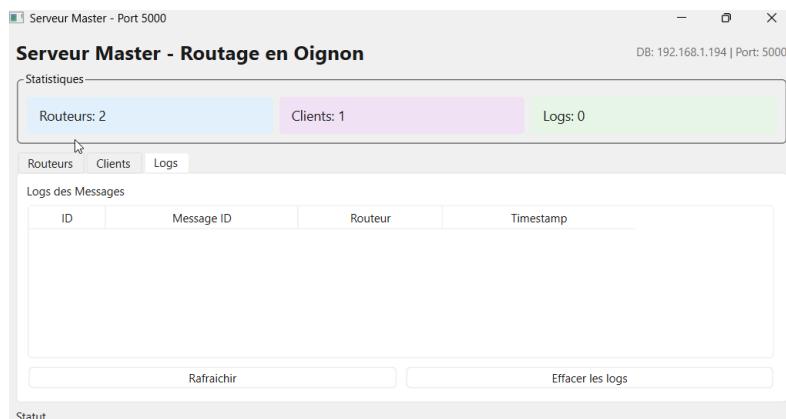
2. Gestion des clients et routeurs :

Supprimer un client ou un routeur : L'interface permet également de retirer un client ou un routeur de la liste en cas de déconnexion ou d'erreur.

Visualisation des connexions : Vous pouvez voir si un client ou un routeur est actuellement actif, ainsi que l'heure de sa dernière connexion.

Logs d'activité :

- Affichage des logs : L'interface affiche un journal des activités en temps réel, qui comprend :
 - Les messages envoyés par les clients.
 - Les connexions établies avec les routeurs.
 - Les erreurs ou alertes (par exemple, si un client ou un routeur ne parvient pas à se connecter correctement).
- Cela permet de suivre l'activité du réseau, de déboguer des problèmes potentiels et de vérifier que le système fonctionne correctement.



4.2. Lancer un Routeur

Accédez au répertoire où se trouve le script [routeur.py](#)

chemin : SAE-onion — > Source —>Routeur

Exécutez le script en utilisant la commande suivante :

```
python routeur.py <NOM_ROUTEUR> <ROUTEUR_PORT> <MASTER_IP>  
<MASTER_PORT>
```

Remplacez **<NOM_ROUTEUR>** par un nom unique pour ce routeur (par exemple **Routeur1**).

Remplacez **<PORT_ROUTEUR>** par le port sur lequel ce routeur écoutera (par exemple **15001**).

Remplacez **<IP_MASTER>** par l'adresse IP du serveur Master

Remplacez **<PORT_MASTER>** par le port sur lequel le serveur Master écoute

4.3. Lancer un Client

Accédez au répertoire où se trouve le script [**client.py**](#).

chemin : SAE-onion — > Source —> Client

Exécutez le script en utilisant la commande suivante :

```
python client.py <NOM_CLIENT> <PORT_CLIENT> <MASTER_IP>  
<MASTER_PORT>
```

Remplacez **<NOM_CLIENT>** par un nom unique pour ce client

Remplacez **<PORT_CLIENT>** par le port sur lequel ce client écoutera les messages entrants

Remplacez **<IP_MASTER>** par l'adresse IP du serveur Master

Remplacez **<PORT_MASTER>** par le port sur lequel le serveur Master écoute

voilà à quoi ressemble l'interface graphique du client

Client amine

Port: 5034 | Master: 192.168.1.24:5000

Enregistre auprès du Master

Envoyer un message

☒ Selectionner un client
☐ Saisir IP et Port

Client:

Nombre de sauts:

Message:

Routeurs disponibles

Selectionner les routeurs a utiliser:

Aucun routeur disponible

Messages recus

L'interface graphique du client permet d'envoyer et de recevoir des messages de manière sécurisée via des routeurs, tout en garantissant l'anonymat grâce au chiffrement en oignon.

1. Envoi de Messages :

Sélectionner un destinataire parmi les clients disponibles ou saisir l'IP et le port manuellement.

☒ Sélectionner un client
☐ Saisir IP et Port

Client:

Envoyer un message

☐ Sélectionner un client
☒ Saisir IP et Port

IP:

Port:

Choisir le **nombre de sauts** (routeurs) pour le message.

Nombre de sauts:

Entrer et envoyer le message via un ou plusieurs routeurs sélectionnés.

2. Réception de Messages :

- Les messages entrants sont affichés dans la zone dédiée

Messages recus

Effacer

3. Gestion des Routeurs :

- **Sélectionner les routeurs** que vous souhaitez utiliser pour le routage du message.

Routeurs disponibles

Selectionner les routeurs a utiliser:

☒ R1 (192.168.1.194:6666)
☐ R2 (192.168.1.194:6667)

Tout selectionner

Tout deselectionner