



**HAL**  
open science

## Rustdesk, une solution open source par auto-hébergement

Mathis Hein

► **To cite this version:**

Mathis Hein. Rustdesk, une solution open source par auto-hébergement. JRES (Journées réseaux de l'enseignement et de la recherche ) 2024, Renater, Dec 2024, Rennes, France. hal-04894008

**HAL Id: hal-04894008**

**<https://hal.science/hal-04894008v1>**

Submitted on 17 Jan 2025

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution - NonCommercial 4.0 International License

# Rustdesk, une solution open source par auto-hébergement

**Mathis HEIN**

Institut PPRIME  
2 Bd des Frères Lumière  
86360 Chasseneuil-du-Poitou

## Résumé

*RustDesk est un logiciel de prise en main à distance open source qui permet de se connecter et contrôler des ordinateurs via le service mis en place.*

*C'est avant tout une alternative sûre et performante aux solutions telles que TeamViewer ou AnyDesk, RustDesk met en avant plusieurs caractéristiques qui le rendent attrayant.*

*RustDesk est particulièrement utile pour le support technique des équipes qui peuvent dépanner des ordinateurs clients sans se déplacer ou pour l'administration système permettant de gérer et maintenir des postes de travail distants.*

*Ses principaux intérêts sont :*

*Open Source : Contrairement à de nombreux concurrents, RustDesk est entièrement open source, ce qui signifie qu'il est gratuit et que son code source est accessible pour inspection, modification et amélioration par la communauté.*

*Sécurité et Confidentialité : RustDesk utilise le chiffrement de bout en bout pour garantir que les connexions sont sécurisées. Les utilisateurs peuvent choisir d'héberger leur propre serveur pour un contrôle total sur leurs données et contrôler qui peut initier la communication.*

*Performances et Réactivité : basé sur le langage Rust, reconnu pour sa performance et sa sécurité, RustDesk offre une expérience utilisateur fluide avec une faible latence.*

*Facilité d'Utilisation : L'interface utilisateur de RustDesk est simple, permettant une prise en main rapide.*

*Compatibilité : Rustdesk offre la possibilité d'accès à distance Multi-OS, il est possible de l'utiliser sur Mac/Windows ou Linux.*

*Rustdesk est une solution sécurisée pour la gestion à distance, offrant une alternative viable aux solutions payantes, tout en garantissant des fonctionnalités intéressantes pour nos besoins.*

*Pour aller plus loin, il est cependant possible d'acheter une licence permettant de disposer d'une console web, un contrôle d'accès, des paramètres centralisés, une authentification à 2 facteurs.*

## Mots-clefs

Contrôle à distance, Open source, RustDesk, Auto-hébergement, Sécurité, Confidentialité, Multi-plateforme, Transfert de fichiers, Connexion rapide, Communauté active, PM2, NSSM, Docker, WAPT, Puppet, Docker Compose, Hole Punching, Prometheus, Grafana

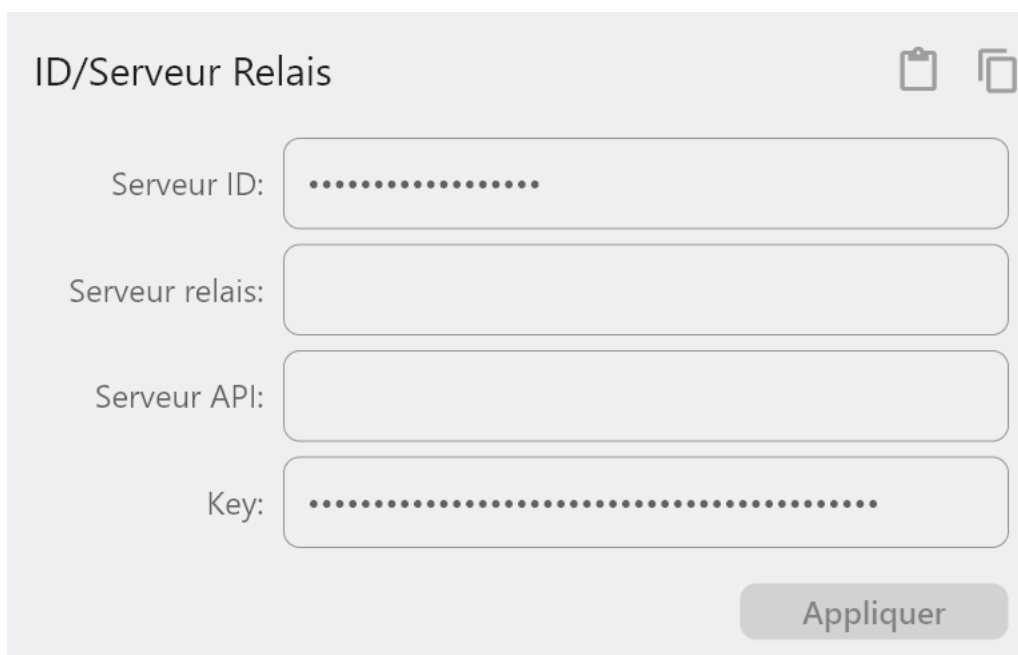
## 1 Introduction

L'utilisation de Rustdesk dans mon laboratoire porte tout son sens de par la disposition d'un parc d'OS hétérogène, l'objectif était ainsi de disposer d'un outil compatible entre les différents OS qui propose l'auto-hébergement et le chiffrement de la communication.

Rustdesk répondait à mes besoins en termes de simplicité d'utilisation et de liberté de fonctionnement.

Les aspects qui m'ont conforté dans l'utilisation de cet outil sont les suivants :

- clé de chiffrement : Cette fonctionnalité revêt une importance cruciale, en permettant le chiffrement des communications pour sécuriser les échanges entre les deux appareils. La solution inclut également l'option de refuser toute communication non chiffrée, ce qui m'a conduit à la mettre en œuvre. Ainsi, seuls les clients possédant la clé peuvent initier une connexion, permettant de distinguer les rôles : les administrateurs ayant la clé peuvent initier les communications, tandis que les utilisateurs sans clé n'en ont pas l'autorisation. Cette « distinction » n'est pas une intégration propre de l'outil, j'ai utilisé une fonctionnalité de Rustdesk pour l'adapter à mon besoin.
- installation silencieuse : Dans mon contexte je déploie Rustdesk à l'aide de WAPT, une solution de déploiement automatisé de logiciels pour Windows, Mac et Linux et qui me permet ainsi par un script Python de rendre l'installation complètement silencieuse. C'est un besoin important dont la solution devait bénéficier pour convenir à mes prérequis.



The image shows a configuration window for Rustdesk with the title "ID/Serveur Relais". It features four input fields stacked vertically:

- Serveur ID:** A text box containing a series of dots, indicating a masked or sensitive value.
- Serveur relais:** An empty text box.
- Serveur API:** An empty text box.
- Key:** A text box containing a long series of dots, representing a masked key.

At the bottom right of the window is a button labeled "Appliquer".

Le client Rustdesk est actuellement déployé sur 55 machines de mon parc, mais l'objectif est bien entendu que l'ensemble des postes de travail en soit équipé. Je le déploie de façon automatisée grâce à la solution WAPT et à l'aide d'un script Python que j'ai créé qui permet à la fois d'installer l'agent, de configurer l'adresse du serveur Rustdesk et d'envoyer l'ID du client dans un fichier texte de mon lecteur partagé où est contenu l'ensemble des ID de mes clients Rustdesk. Cette méthode me permet d'avoir une vision claire des différents ID sans pour autant disposer d'une licence payante pour les visualiser sur un tableau de bord.

## 2 Détails techniques de RustDesk

Rentrons maintenant un petit peu plus dans le détail de l'utilisation de Rustdesk et de son fonctionnement interne.

### 2.1 Architecture

RustDesk se base sur une architecture client-serveur, où deux types de serveurs peuvent être utilisés pour faciliter les connexions :

**hbbs:** C'est le serveur principal, responsable de la gestion des connexions des clients. Il gère la négociation initiale de connexion entre le client et l'hôte distant. Ce serveur permet de remplacer l'utilisation des serveurs publics fournis par RustDesk, offrant ainsi une solution auto hébergée plus sécurisée.

**hbbr:** Ce serveur agit comme un relais pour transmettre le trafic lorsque la connexion directe entre le client et l'hôte est impossible (par exemple, en raison de restrictions NAT ou de pare-feu).

Dans une majorité de cas, le serveur HBRR n'est jamais utilisé, car la connexion directe est bien souvent possible (Il n'est pas obligatoire d'avoir le serveur HBRR pour le fonctionnement de Rustdesk)

### 2.2 Protocoles

- Le serveur hbbs écoute sur les ports suivants :
  - 21115 (TCP) : Utilisé pour tester le type de NAT
  - 21116 (TCP/UDP) : UDP pour l'enregistrement d'ID et le service heartbeat
    - TCP : Pour le service de connexion
- Le serveur hbbr écoute sur le port suivant :
  - 21117 (TCP) : service relais

Un port additionnel est utilisé pour ceux disposant de la licence pro de Rustdesk :

- 21114 : Port utilisé pour la console Web

### 3 Méthode installation serveur

Dans mon contexte d'installation, Rustdesk n'est pas installé à la main, j'utilise une solution automatisée de gestion des infrastructures. L'intérêt est de m'assurer que le service est configuré de manière conforme, avec un versioning et une mise à jour rapide. Rustdesk est pour ma part installé sur une Debian 12, c'est une question de préférence, mais il est tout à fait possible de l'installer sur un Windows Server par exemple. Tout est centralisé sous un Git avec une branche de test et une autre de production.

J'utilise la branche de test pour qualifier une nouvelle version de serveur avec un possible changement de configuration ensuite. Tout est ainsi déployé sur un serveur de test et une fois la validation du bon fonctionnement de la nouvelle version, la branche de test est envoyée sur la branche de production pour ainsi procéder au déploiement en production.

#### 3.1 Puppet

Puppet est un outil de gestion de configuration largement utilisé pour automatiser le déploiement, la gestion et la maintenance des infrastructures informatiques. Il permet aux administrateurs système de définir des états souhaités pour les ressources sur leurs serveurs (tels que des fichiers, services, ou paquets logiciels) à travers des manifests écrits en langage déclaratif. Une fois ces états définis, Puppet s'assure que les systèmes concernés respectent ces configurations en les appliquant automatiquement et en maintenant les systèmes à jour, minimisant ainsi les erreurs humaines et les écarts de configuration.

Puppet fonctionne selon une approche client-serveur, où un Puppet Master central gère et contrôle plusieurs agents installés sur les machines à administrer, dans cette approche Rustdesk sera ainsi un serveur administré par Puppet.

#### 3.2 Installation standard serveur

L'installation de Rustdesk est relativement facile, voici concrètement la procédure d'installation :

- récupérer le package .deb depuis le GitHub de Rustdesk sur la page suivante : <https://github.com/rustdesk/rustdesk-server/releases/latest>
- choisir le package hbbs et hbbr;
- installer ainsi les .deb en `apt install <nomfichier>.deb` ;
- Modifier l'exécution du service pour n'autoriser que la clé publique : `systemctl edit --full rustdesk-hbbs` puis modifier la ligne `ExecStart` en ajoutant à la fin « `-k _` ».

### 3.3 Installation sous Puppet

Comme présenté le déploiement de l'infrastructure Rustdesk est dans mon contexte déployé depuis un environnement Puppet, il suffit ainsi de s'appuyer sur la partie de l'installation standard pour le transformer sous une version Puppet avec quelques modifications. Par exemple dans une situation où le serveur doit être réinstallé, il suffira de déployer Puppet sur le poste puis de déployer la conf Rustdesk, sauf que dans ces conditions la clé privée et publique seront modifiées et il faut ainsi dès le départ stocker dans une partie « files » la clé privée et publique afin d'être en mesure de la recopier une fois l'installation faite. Puppet vérifie si le fichier est identique à celui présent dans l'environnement. Puppet déclenche uniquement l'action si le fichier n'existe pas ou que le contenu diffère.

Voici une partie du manifest pour le serveur de production :

```
class pprime_rustdesk::server(  
  String $version = "1.1.8"  
)  
{  
  # Copie du .deb hbbw présent dans /home/puppetlabs/private du serveur puppet  
  file { 'rustdesk_hbbw':  
    path => '/root/rustdesk_hbbw.deb',  
    ensure => 'file',  
    source => "puppet:///private/rustdesk-server-hbbw-${version}_amd64.deb",  
    owner => 'root',  
    group => 'root',  
    mode => '0755',  
    notify => Exec['rustdesk_install_hbbw'],  
  }  
  # Copie du .deb hbbs présent dans /home/puppetlabs/private du serveur puppet  
  file { 'rustdesk_hbbs':  
    path => '/root/rustdesk_hbbs.deb',  
    ensure => 'file',  
    source => "puppet:///private/rustdesk-server-hbbs-${version}_amd64.deb",  
    owner => 'root',  
    group => 'root',  
    mode => '0755',  
    notify => Exec['rustdesk_install_hbbs'],  
  }  
  # Installation du Service HBBS -> Service Principal de fonctionnement RustDesk  
  exec { 'rustdesk_install_hbbs':  
    command => "apt install -f /root/rustdesk_hbbs.deb",  
    path => ['/bin', '/sbin', '/usr/bin', '/usr/sbin'],  
    refreshonly => true,  
  }  
}
```

Figure 1 Fichier Manifest Puppet Rustdesk

String \$version : Cette variable me permet de définir automatiquement sous quelle version le serveur Rustdesk doit tourner, c'est un paramètre à changer en premier lieu sur le manifest du serveur de test avant modification sur le serveur de production.

Dans le cas où l'archive est présente sur la plateforme Git de Rustdesk, ici en l'occurrence pour la classe fille nommée « rustdesk\_hbbw » et « rustdesk\_hbbs », nous allons directement chercher depuis un chemin présent sur le serveur Puppet les archives.

J'entrepose ainsi dans ce dossier « private » uniquement les archives en phase de test et de production.

La classe exec « rustdesk\_install\_hbbs » me permet ainsi depuis le shell d'installer le service rustdesk\_hbbs dans un premier temps.

### 3.4 Schématisation de la communication

Pour expliquer au mieux la communication P2P entre 2 postes et l'utilisation de HBBS et de HBBR voici dans un premier temps un schéma simplifié :

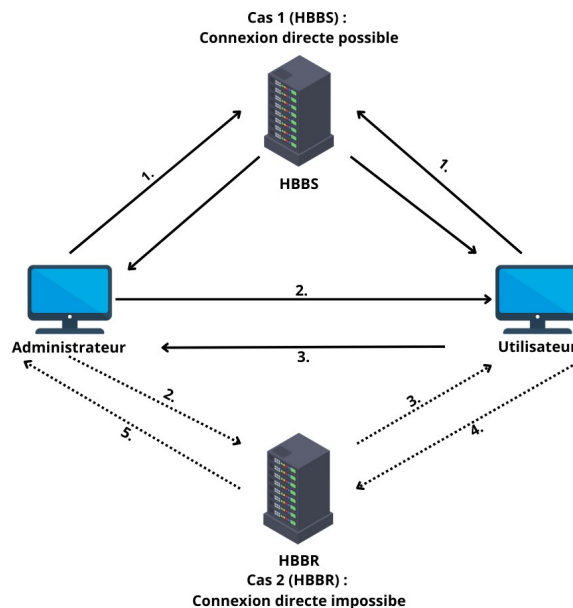


Figure 2: schématisation d'une communication

- [1.] L'Administrateur demande l'accès au PC distant (Utilisateur) en utilisant son ID. Le serveur ID (HBBS) répond à la demande en fournissant l'ID unique de chaque client ainsi que la clé publique aux deux appareils.
- [2.] Dans le cas 1 si une connexion directe est possible, grâce au TCP hole punching (une technique permettant aux appareils situés derrière des routeurs NAT de se connecter en peer-to-peer), les deux appareils (Administrateur et Utilisateur) peuvent communiquer directement sans passer par le serveur relais (HBBR). Le PC distant affiche son écran et permet au client de prendre le contrôle (uniquement si le mot de passe est correct ou si le PC distant accepte manuellement l'accès).  
Échec de la connexion directe (Cas 2) : Si le hole punching échoue, les actions et flux de données passeront par le serveur relais (HBBR) pour permettre la communication.
- [3.] Contrôle à distance (Cas 1) : Dans le cas d'une connexion directe, une fois la demande de contrôle acceptée, l'Administrateur prend le contrôle du PC distant. Les flux vidéo et audio sont directement envoyés entre les deux appareils, permettant un contrôle fluide.

Contrôle indirect (Cas 2) : Dans le cas où la connexion passe par le serveur relais (HBBR), les actions effectuées par l'Administrateur sont transmises au HBBR..

[4.] Transmission via le serveur relais (Cas 2) : Dans ce cas indirect, le flux vidéo et audio du PC distant est d'abord envoyé au serveur relais (HBBR), plutôt que directement au client.

[5.] Relais vers le client (Cas 2) : Le serveur relais (HBBR) transmet ensuite le flux vidéo et audio à l'Administrateur, permettant à celui-ci de contrôler le PC distant, même si la connexion directe n'a pas pu être établie..

## 4 Docker

Docker est une plateforme open source qui facilite le développement, le déploiement et l'exécution d'applications à l'aide de conteneurs. Les conteneurs sont des environnements isolés qui regroupent tout le nécessaire pour faire fonctionner une application, y compris le code, les bibliothèques et les dépendances. Cela permet de mettre en place un environnement rapidement grâce à des images et des configurations par Docker Compose par exemple.

Dans le contexte de la gestion des infrastructures, Puppet a permis de standardiser et d'automatiser les configurations nécessaires au déploiement de RustDesk. Cependant, pour aller plus loin en termes de rapidité de déploiement et de portabilité, Docker est une solution intéressante complémentaire ou de substitut à un environnement Puppet.

Docker Compose est utilisé pour déployer plusieurs conteneurs à l'aide d'un seul fichier YAML.

Nous verrons dans le chapitre 3.2 l'utilisation de Docker Compose pour Rustdesk.

Docker est compatible sur une multitude de distributions Linux telles que CentOS / Debian / Ubuntu / Fedora.

### 4.1 Configuration Rustdesk sous Docker

Nous allons maintenant voir comment installer le serveur Rustdesk en utilisant Docker.

Avant de créer le conteneur, voici un script géré par Docker pour l'installation :

```
curl -fsSL https://get.docker.com -o get-docker.sh
sh get-docker.sh
```

```
sudo docker image pull rustdesk/rustdesk-server
```

```
sudo docker run --name hbbs -v ./data:/root -td --net=host --restart unless-stopped
rustdesk/rustdesk-server hbbs
```



```
sudo docker run --name hbbr -v ./data:/root -td --net=host --restart unless-stopped
rustdesk/rustdesk-server hbbr
```

## 4.2 Configuration Docker par Docker Compose

Pour installer les conteneurs Rustdesk avec Docker Compose il faut disposer du plugin :

```
sudo apt-get update
sudo apt-get install docker-compose-plugin
```

### 4.2.1 Fichier Docker Compose

Afin d'avoir un aperçu clair de la configuration Docker, je recommande de créer un dossier « Docker » dans lequel vous allez créer un dossier « Rustdesk » puis à l'intérieur le fichier que vous nommez « docker-compose.yml »

Voici le contenu de ce fichier :

```
services:
  hbbs:
    container_name: hbbs
    image: rustdesk/rustdesk-server:latest
    command: hbbs
    volumes:
      - ./data:/root
    network_mode: "host"

    depends_on:
      - hbbr
    restart: unless-stopped

  hbbr:
    container_name: hbbr
    image: rustdesk/rustdesk-server:latest
    command: hbbr
    volumes:
      - ./data:/root
    network_mode: "host"
    restart: unless-stopped
```

### 4.2.2 Lancement des conteneurs par Docker Compose

Il reste plus qu'à lancer les conteneurs grâce à la commande suivante :

```
docker-compose up -d
```

Lorsqu'on lance des conteneurs avec Docker Compose, plusieurs options s'offrent à nous pour surveiller et diagnostiquer l'état des conteneurs :

### **Visibilité des Sessions Actives :**

En exécutant `docker ps`, il est possible d'obtenir des informations de base sur l'état des conteneurs actifs sur le serveur, comme leur statut (en cours d'exécution, arrêté, etc.).

### **Outils de Diagnostic :**

Docker propose des commandes telles que `docker logs [nom_du_conteneur]` pour accéder aux journaux générés par les applications, ce qui est utile pour diagnostiquer des erreurs ou anomalies. De plus, la commande `docker stats` permet de surveiller la consommation de ressources (CPU, mémoire) en temps réel pour chaque conteneur, offrant ainsi une vue d'ensemble des performances.

### **Outils de Dépannage :**

Pour effectuer un dépannage plus approfondi, `docker exec -it [nom_du_conteneur] bash` permet d'accéder directement au shell du conteneur. Cette approche est utile pour examiner les configurations, exécuter des diagnostics au sein du conteneur ou consulter des logs supplémentaires. Pour un monitoring continu, des solutions comme Prometheus et Grafana permettent de visualiser les métriques des conteneurs.

## **5 Configuration Client**

Dans cette partie nous allons décrire en détail la configuration des clients sur les différentes plateformes, Windows, Mac et Linux.

### **5.1 Client Windows**

La configuration du client Rustdesk sur Windows est un processus simple et direct. Voici les étapes à suivre :

#### **5.1.1 Téléchargement et Installation :**

Rendez-vous sur le site officiel de Rustdesk pour télécharger la dernière version du client Windows.

Exécutez le fichier téléchargé et suivez les instructions de l'assistant d'installation.

Lien du téléchargement : <https://github.com/rustdesk/rustdesk/releases>

# The Fast Open-Source Remote Access and Support Software

Switch from TeamViewer, AnyDesk, and Splashtop to RustDesk for a secure and reliable remote desktop experience with your own self-hosted servers.

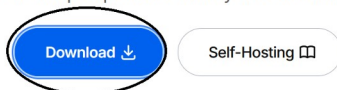


Figure 3 Site officiel rustdesk

## 5.1.2 Configuration initiale :

Après installation, ouvrez l'application.

## 5.1.3 Paramètres Réseau :

Accédez aux paramètres réseaux pour configurer le serveur signal (HBBS) et la clé publique pour chiffrer la communication. (Paramètres réseaux accessibles depuis le cercle noir)

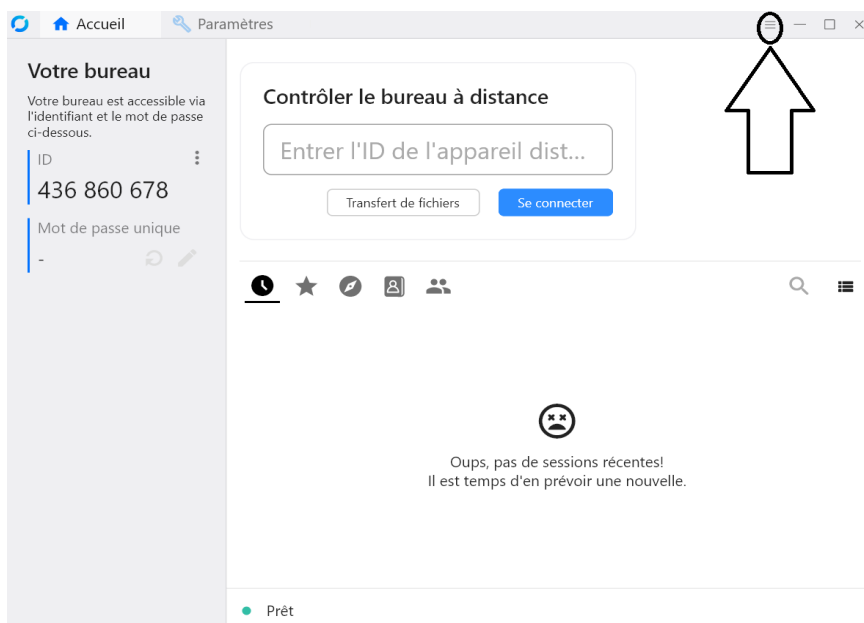


Figure 4 : Capture d'écran de l'accueil du client

Pour rappel la clé publique du serveur Rustdesk se trouve dans le chemin suivant (/var/lib/rustdesk-server/id\_ed25519.pub)

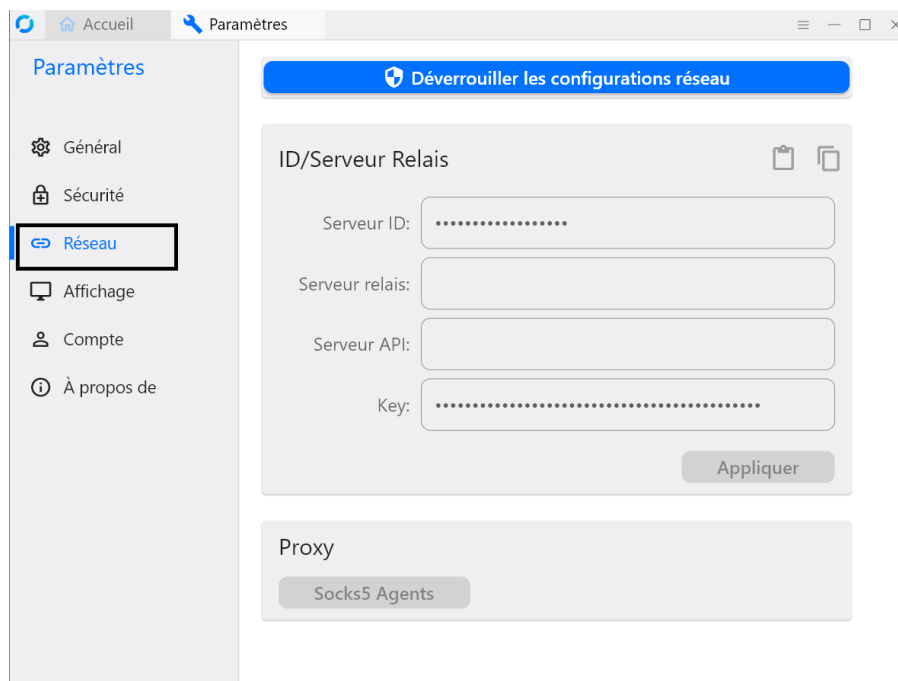


Figure 5 Paramètres Réseaux du client sous Windows

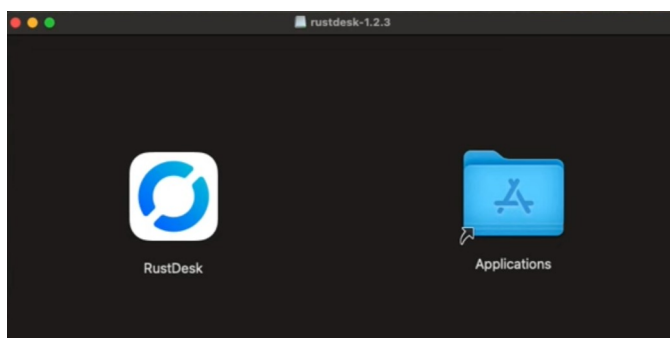
## 5.2 Client Mac

Pour les utilisateurs de Mac, la configuration de Rustdesk est également intuitive. Voici comment procéder :

### 5.2.1 Téléchargement et Installation :

Téléchargez le client Rustdesk pour Mac à partir du dépôt git.  
(<https://github.com/rustdesk/rustdesk/releases>)

Ouvrez le fichier .dmg et glissez l'application dans le dossier Applications.



## 5.2.2 Lancement et Configuration :

Lancez Rustdesk

Configurez les paramètres réseau à l'identique de la partie 5.1.3

## 5.2.3 Préférences de Sécurité :

Pour le client Mac il faut également modifier des paramètres de sécurité.

Si vous revenez sur la partie accueil du client, vous avez ces encadrés :

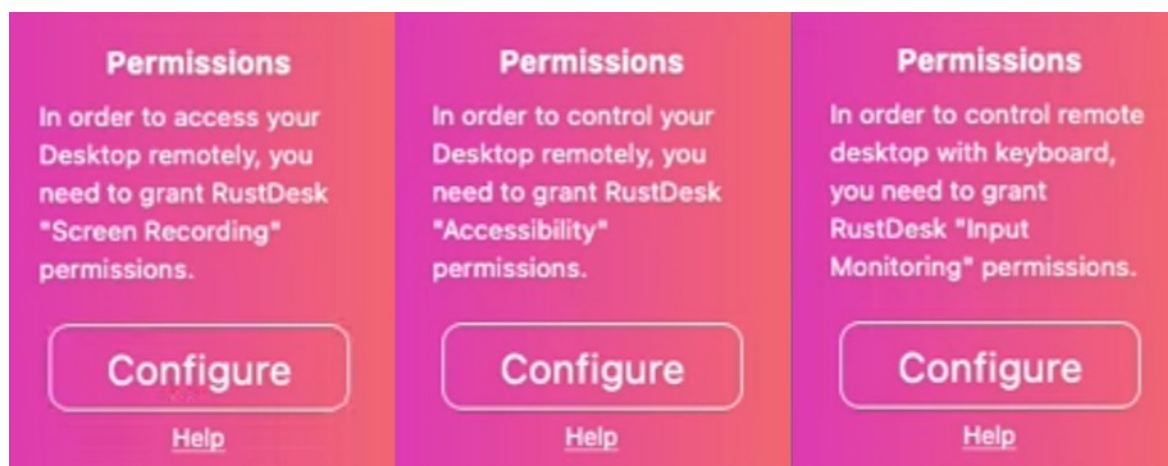


Figure 6: Permission sur mac

Pour donner l'ensemble des permissions, il faut cliquer sur « Configure » puis « Open System Settings » depuis l'onglet qui s'ouvrira et activer la permission sur la nouvelle page.

L'encadré apparaîtra 3 fois au total pour 3 permissions « Screen Recording Permissions » « Accessibility » et « Input Monitoring »

## 5.3 Client Linux

La configuration de Rustdesk sur Linux peut varier légèrement selon la distribution utilisée, mais les principes restent les mêmes.

### 5.3.1 Installation :

Téléchargez le paquet approprié pour votre distribution (DEB pour Debian/Ubuntu, RPM pour Fedora/CentOS, etc.).

Pour rappel voici le lien vers la page des paquets : <https://github.com/rustdesk/rustdesk/releases/>

Utilisez la commande appropriée pour installer Rustdesk. Par exemple :

```
sudo dpkg -i rustdesk_<version>.deb
```

### 5.3.2 Configuration de l'Application :

Lancez Rustdesk

Comme pour les autres systèmes, configurez les paramètres réseau.

## 6 Conclusion

Rustdesk représente une solution intéressante dans le domaine de l'accès à distance open source. Grâce à son architecture sécurisée, sa facilité d'utilisation, et ses possibilités de personnalisation, RustDesk répond aux besoins d'un parc de n'importe quelle taille.

Les points détaillés ici illustrent comment, avec une configuration simple, RustDesk peut être intégré efficacement dans divers environnements.

Cependant, comme pour tout outil, il est essentiel de suivre les mises à jour, les bonnes pratiques en matière de sécurité et d'optimiser les paramètres en fonction de chaque cas d'utilisation.

Ainsi, que vous soyez à la recherche d'une alternative à des solutions propriétaires, RustDesk mérite une attention particulière. En investissant du temps pour comprendre ses fonctionnalités et ses options de configuration avancées, vous pourrez tirer parti de tout son potentiel pour une gestion de l'accès à distance sécurisée et personnalisée tout en l'utilisant dans un environnement Puppet ou Docker par exemple.

# Annexe

## Bibliographie

- [1] Site officiel Rustdesk ; <https://rustdesk.com/> .
- [1] Documentation Rustdesk ; <https://rustdesk.com/docs/en/> .
- [2] Github Rustdesk ; <https://github.com/rustdesk/doc.rustdesk.com> .
- [3] FAQ Rustdesk ; <https://github.com/rustdesk/rustdesk/wiki/FAQ> .