RustDesk

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.

Présentation

Qu’est-ce que RustDesk ?

RustDesk est un logiciel open source qui permet de prendre le contrôle à distance d’un ordinateur, un peu comme TeamViewer ou AnyDesk. Il est conçu pour être simple, rapide, et surtout respectueux de la vie privée, car il peut être auto-hébergé, c’est-à-dire installé sur vos propres serveurs.

RustDesk permet de se connecter à distance à un poste de travail, de partager l’écran, d’échanger des fichiers, et d’assister un utilisateur, tout en offrant une sécurité renforcée grâce au chiffrement des données.

À quoi sert RustDesk ?

RustDesk sert principalement à accéder à distance à un ordinateur pour le gérer, le dépanner, ou collaborer. C’est un outil précieux pour les équipes techniques qui doivent intervenir sur des postes distants, ou pour les particuliers qui veulent accéder à leur ordinateur personnel depuis l’extérieur.

Il remplace les solutions propriétaires en offrant la possibilité de garder le contrôle total sur les données en auto-hébergeant son propre serveur de connexion.

Comment fonctionne RustDesk ?

RustDesk utilise un système client-serveur. Les machines clientes communiquent via un serveur de rendez-vous (appelé « rendez-vous server ») et un serveur relais, qui facilitent la connexion même derrière des routeurs ou pare-feux.

Les données échangées sont chiffrées de bout en bout, ce qui garantit que personne ne peut intercepter les informations ou prendre le contrôle sans autorisation.

Pourquoi utiliser RustDesk ?

RustDesk est une solution idéale si vous cherchez un outil de contrôle à distance simple à déployer, sans dépendre de serveurs tiers. Son auto-hébergement permet de protéger votre vie privée et d’éviter que des données sensibles ne transitent par des services externes.

Il est compatible avec plusieurs systèmes d’exploitation, léger, et propose une interface facile à prendre en main.

Cas d’usage courants

RustDesk est utilisé pour :

- Assister des utilisateurs à distance en entreprise ou à domicile,
- Administrer des serveurs ou postes distants,
- Partager rapidement des fichiers entre machines,
- Travailler à distance en gardant la maîtrise des données,
- Remplacer des solutions propriétaires coûteuses ou fermées.

En résumé

RustDesk est une alternative open source, simple et sécurisée, au contrôle à distance classique. Son principal atout est la possibilité d’auto-héberger ses serveurs, garantissant ainsi confidentialité et autonomie.

Installation

Fichier docker-compose

Dans cette procédure, nous allons utiliser **Docker**. Une documentation préalable à ce sujet est disponible ici.
Commençons par créer un fichier `docker-compose.yaml`, adapté à nos besoins :

```
services:
  # Service hbbs : serveur de rendez-vous (ID Server)
  hbbs:
    container_name: hbbs  # Nom explicite du conteneur
    image: rustdesk/rustdesk-server:latest  # Image officielle du serveur RustDesk
    command: hbbs  # Lance le composant "hbbs" (Heartbeat Broker Server)
    volumes:
      - ./data:/root  # Monte le dossier local "data" dans /root du conteneur pour conserver
les clés et paramètres
    network_mode: "host"  # Utilise directement le réseau de l'hôte (nécessaire pour les ports
UDP/TCP fixes)
```

```
    depends_on:
      - hbbr  # Démarre uniquement si le serveur relais (hbbr) est déjà en cours d'exécution
    restart: unless-stopped  # Redémarrage automatique sauf si le conteneur est arrêté manuellement

  # Service hbbr : serveur de relais (Relay Server)
  hbbr:
    container_name: hbbr  # Nom du conteneur
    image: rustdesk/rustdesk-server:latest  # Utilise la même image que hbbs
    command: hbbr  # Lance le composant "hbbr" (Relay Server)
    volumes:
      - ./data:/root  # Partage le même dossier que hbbs (utile pour les certificats et clés)
    network_mode: "host"  # Nécessaire pour exposer les ports réseau sans restriction
    restart: unless-stopped  # Redémarrage automatique si le conteneur s'arrête de manière imprévue
```

Dans ce fichier, nous n'avons spécifié **aucun port explicitement**, car nous utilisons le mode réseau `host`. En revanche, il est important de connaître les ports utilisés par RustDesk, notamment si un **pare-feu** est en place ou si le serveur doit être **accessible depuis l'extérieur**.

Ports utilisés par RustDesk

Port	Protocole	Description
21115	TCP	Test du type de NAT
21116	UDP	Enregistrement de l'ID et service Heartbeat
21116	TCP	Connexions entrantes
21117	TCP	Service de relais
21118	TCP	Client Web (non obligatoire)
21119	TCP	Client Web (non obligatoire)

Démarrage

Une fois le fichier `docker-compose.yaml` créé, nous pouvons **lancer RustDesk** avec la commande suivante :

```
docker compose up -d
```

Récupération de la clé publique

Il reste une dernière étape essentielle : récupérer la **clé publique** générée par le serveur. Celle-ci est nécessaire pour que les clients puissent établir une connexion chiffrée avec le serveur :

```
docker cp <container-id>:/id_ed25519.pub <destination>
```

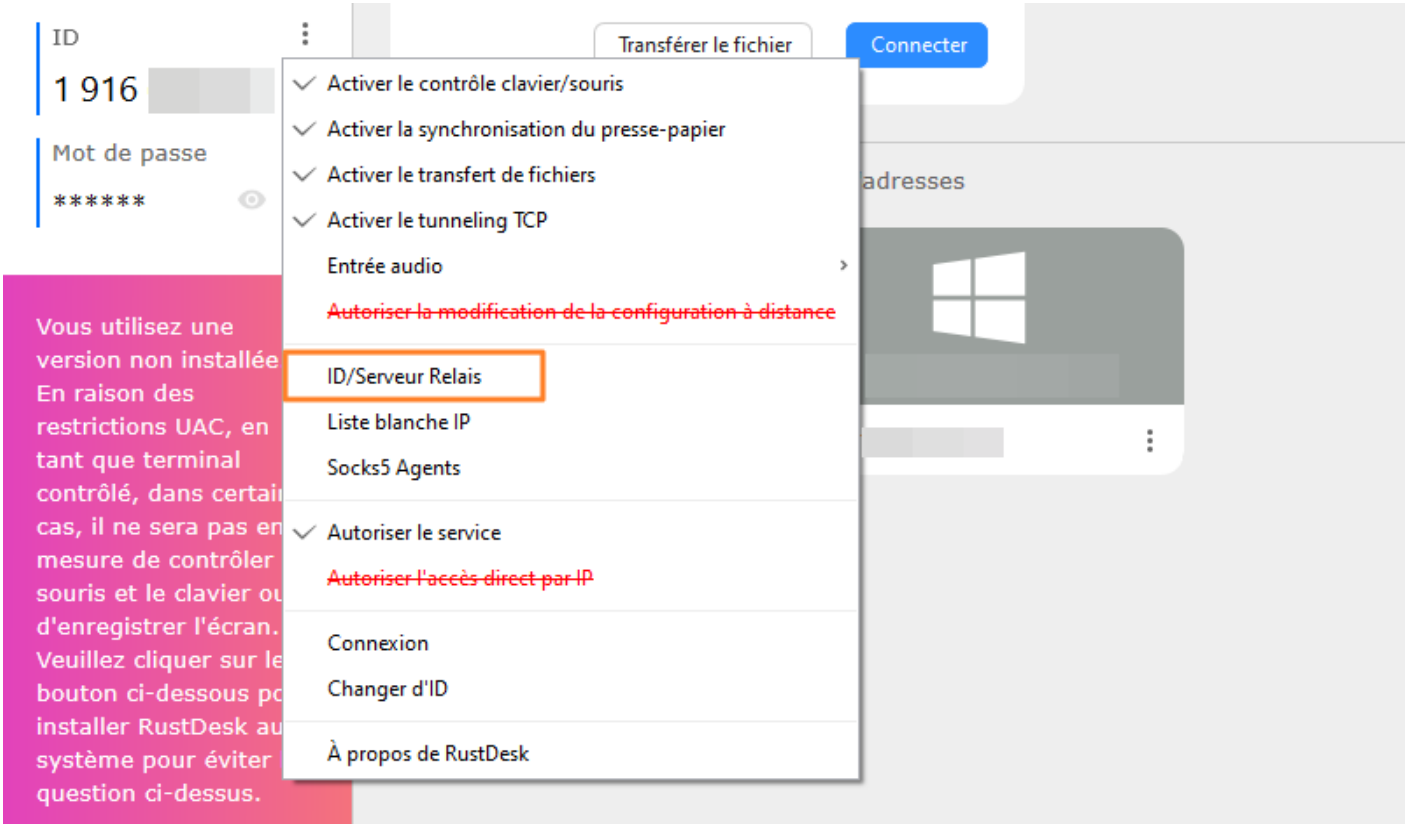
> Remarque : le chemin ou le nom du conteneur peut varier si vous avez modifié les paramètres de base du fichier docker-compose.

Prise en main à distance

Configuration client

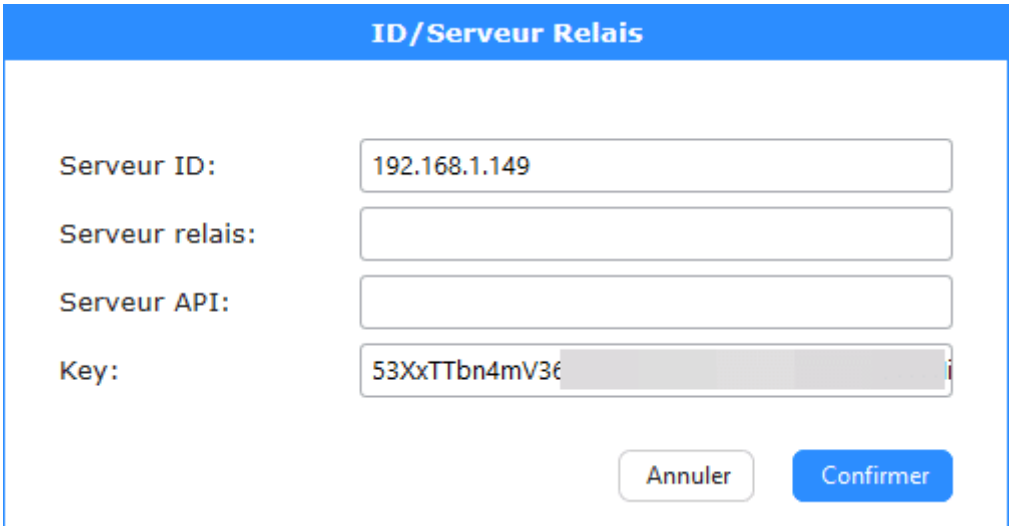
Pour tester RustDesk en passant par notre serveur, nous avons besoin de deux clients avec RustDesk. Pour rappel, le client RustDesk est téléchargeable directement sur GitHub. Sous Windows, il se présente sous la forme d'un exécutable. Il peut-être installé ou simplement utilisé en mode portable.

Une fois lancé, il faut que l'on configure le client RustDesk pour lui dire d'utiliser notre serveur auto-hébergé. Pour cela, **cliquez sur le menu avec les trois points** à côté de l'ID, puis sur l'option "**ID/Serveur Relais**".



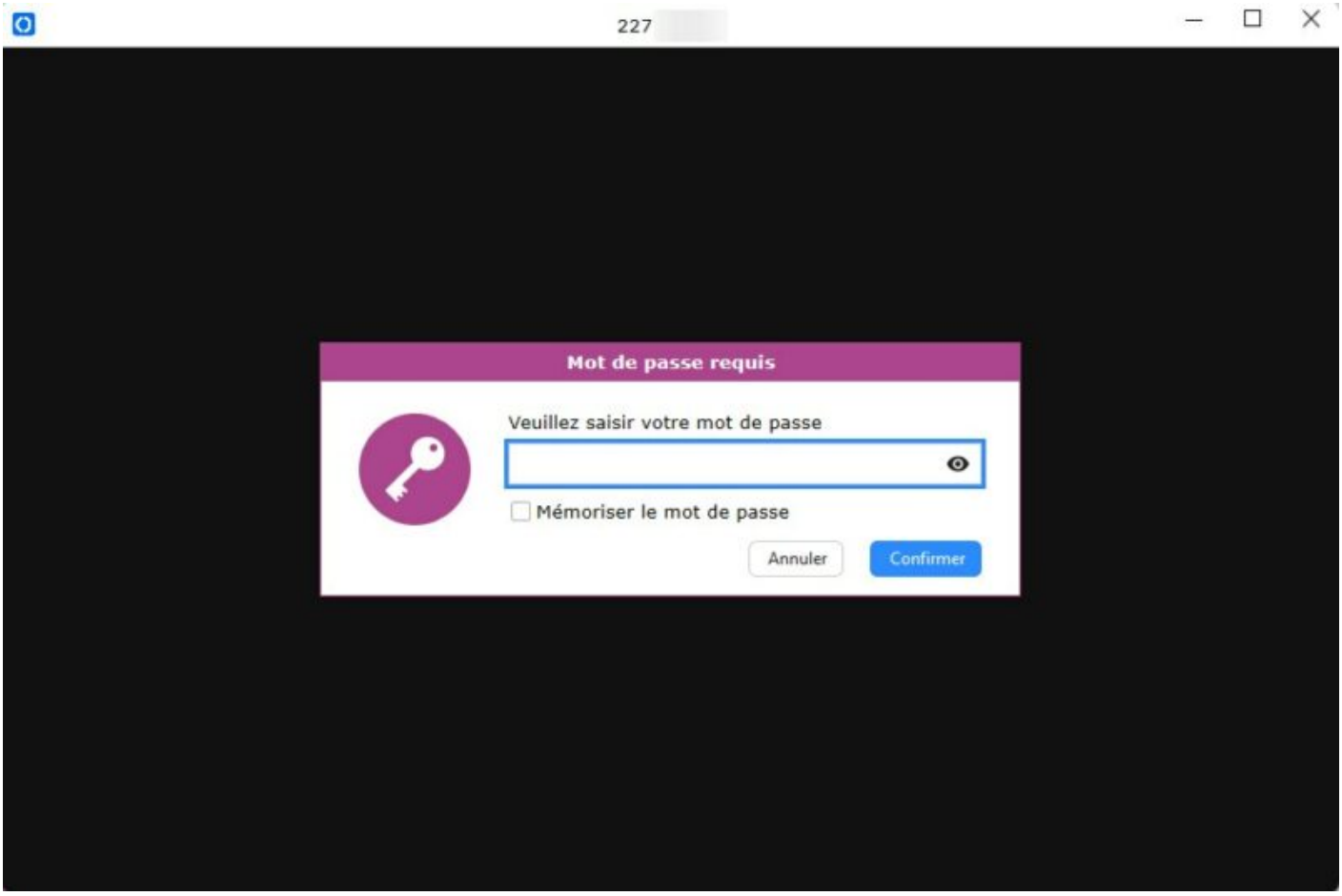
Renseignez simplement le champ "**Serveur ID**" en indiquant l'adresse IP du serveur à contacter (ou le nom de domaine DNS). Cela signifie que vous devez indiquer l'adresse IP privée ou publique selon votre configuration.

Vous devez également renseigner le champ "**Key**" afin d'indiquer **la clé publique générée par votre serveur RustDesk**. Sans cela, la connexion ne sera pas chiffrée, mais pourra fonctionner. Quand c'est fait, cliquez sur "**Confirmer**".

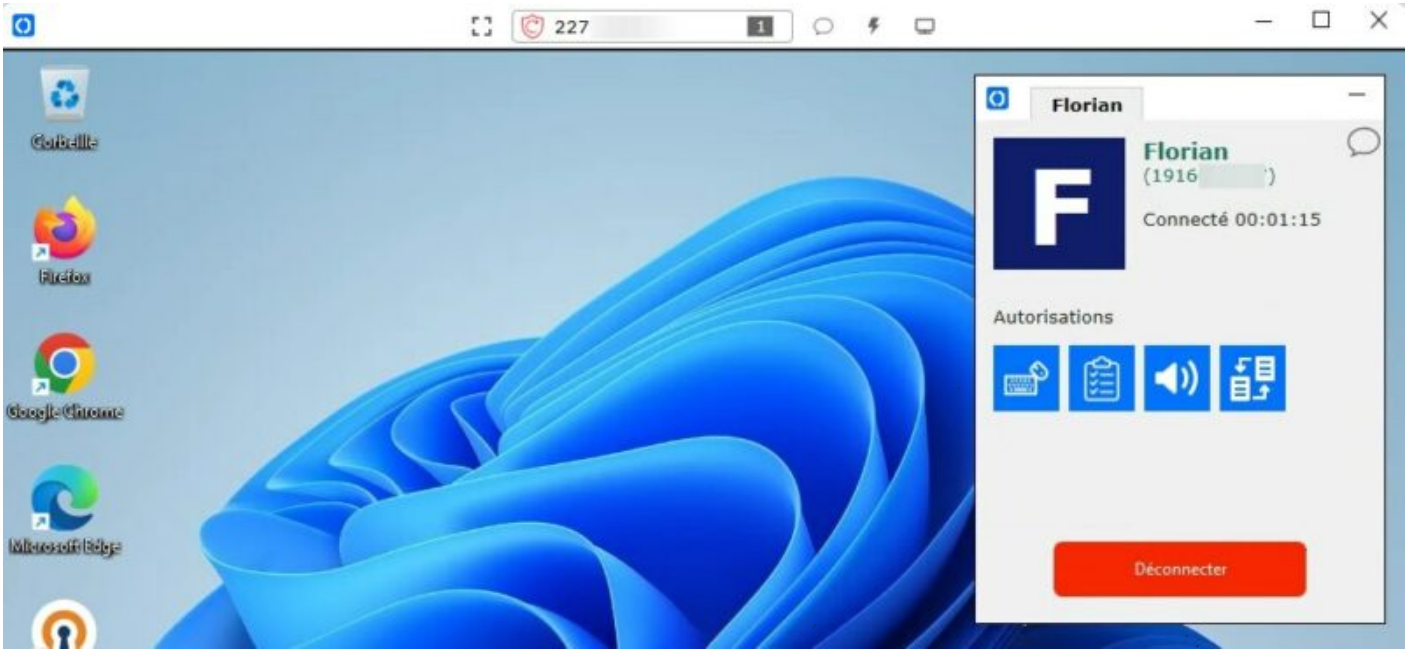


Effectuez la configuration du client RustDesk sur les deux hôtes. Ensuite, vous pouvez tester la connexion pour se connecter d'un hôte A vers un hôte B. Normalement, le client RustDesk doit afficher le statut "**Prêt**" s'il s'est bien connecté à votre serveur.

Saisissez l'ID de l'hôte distant dans le client RustDesk et validez... La connexion va être établie, et il faudra saisir le mot de passe pour se connecter (accessible depuis l'interface de l'hôte distant). L'alternative consiste à accepter la connexion manuellement, ce qui sera possible si un utilisateur est devant l'ordinateur. D'ailleurs, **il vaut mieux utiliser ce principe lorsque l'on se connecte sur un poste d'un utilisateur afin d'avoir son approbation manuelle**.



Lorsque l'on se connecte sans avoir renseigné la clé publique dans les paramètres, on voit qu'un bouclier rouge s'affiche en haut de l'interface. Cela indique que la connexion n'est pas sécurisée.



À l'inverse, quand le client est correctement configuré, **la connexion est bien sécurisée et chiffrée**.

Voilà, vous n'avez plus qu'à profiter de RustDesk !