27 - Rustdesk

RustDesk — Serveur auto-hébergé (hbbs/hbbr)

Qu'est-ce que c'est

RustDesk est un logiciel de bureau à distance open-source (alternative à TeamViewer/AnyDesk). Il repose sur deux briques serveur :

- **hbbs** (ID/rendezvous server) — enregistre les IDs des clients, gère le heartbeat, tente d'établir une connexion P2P directe.
- **hbbr** (relay server) — relaie le flux quand le P2P direct échoue (NAT symétrique, pare-feu strict, etc.).

Auto-héberger ces deux services évite de dépendre des serveurs publics RustDesk (rendezvous/relais tiers) : tout le trafic de contrôle passe par une infrastructure qu'on maîtrise.

Poids du service Docker

Élément	Valeur
Image `rustdesk/rustdesk-server:latest` (amd64)	**5,61 Mo** compressée
Image (arm64)	5,2 Mo
Image (armv7)	5,04 Mo
Nombre de containers	2 (hbbs + hbbr)
RAM au repos	quelques dizaines de Mo par container (binaires Rust natifs, pas de VM/interpréteur)
CPU au repos	négligeable

Élément	Valeur
CPU/RAM en charge	dépend du nombre de sessions actives relayées par hbbr (le flux vidéo transite par lui si pas de P2P direct) — reste très léger pour un usage perso/familial (quelques sessions simultanées max)

Pour comparaison, c'est très inférieur au poids des autres containers déjà en place sur le VPS Jux (BookStack/MariaDB, Immich, etc.).

Ports requis (conditions réseau)

Port	Protocole	Service	Rôle
21115	TCP	hbbs	Test de type NAT
21116	TCP + UDP	hbbs	UDP = enregistrement ID + heartbeat / TCP = hole punching + connexion
21117	TCP	hbbr	Service de relais
21118	TCP	hbbs	Client web (optionnel)
21119	TCP	hbbr	Client web (optionnel)
21114	TCP	hbbs	Console web — **Pro uniquement**, pas utile en OSS

Si le client web n'est pas utilisé, 21118/21119 peuvent rester fermés (recommandé : ces deux ports font confiance aux en-têtes `X-Real-IP` / `X-Forwarded-For` sans les valider, donc à n'exposer que derrière un reverse proxy qui les fixe lui-même, jamais en direct).

docker-compose.yml (référence officielle RustDesk)

```
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
```

- `network_mode: host` est la méthode recommandée sous Linux : hbbs/hbbr voient l'IP réelle du client au lieu de l'IP interne du container. Fonctionne uniquement sur Linux (donc OK sur le VPS Debian, pas sur un NAS DSM qui a ses propres contraintes réseau — voir plus bas).
- Alternative sans host networking (mapping de ports classique, pour Portainer/stack isolée) :

```
services:
  hbbs:
    container_name: hbbs
    image: rustdesk/rustdesk-server:latest
    command: hbbs -r rustdesk.exemple.com:21117
    ports:
      - "21115:21115"
      - "21116:21116"
      - "21116:21116/udp"
      - "21118:21118"
    volumes:
      - ./data:/root
    depends_on:
      - hbbr
    restart: unless-stopped

  hbbr:
    container_name: hbbr
    image: rustdesk/rustdesk-server:latest
```

```
    command: hbbr
    ports:
      - "21117:21117"
      - "21119:21119"
    volumes:
      - ./data:/root
    restart: unless-stopped
```

Variable d'environnement utile : `ALWAYS_USE_RELAY=Y` sur hbbs pour forcer systématiquement le passage par le relais (utile si on veut masquer les IP réelles des clients, au prix d'un peu plus de charge sur hbbr).

Clé de chiffrement et configuration client

- Au premier démarrage, hbbs génère une paire de clés dans le volume `./data` : le fichier **`id_ed25519.pub`** contient la clé publique à distribuer aux clients.
- Configuration côté client RustDesk (menu ⋮ → **Réseau**, déverrouiller) :
  - **ID Server** : IP ou domaine du VPS (ex. `vps.juxjux.ovh` ou l'IP directe)
  - **Relay Server** : peut rester vide (déduit automatiquement, sinon `IP:21117`)
  - **Key** : contenu de `id_ed25519.pub`
- Export/import de config possible entre clients (bouton "Export Server Config"), pratique pour déployer sur plusieurs machines (PC Windows, Ubuntu, etc.) sans ressaisir.

VPS Jux vs NAS Synology sasnexte — analyse

RustDesk propose officiellement un **guide d'installation Synology** (DSM 6 et DSM 7.2/Container Manager), donc l'hébergement NAS est une voie supportée en soi — la question n'est pas la faisabilité technique mais la **joignabilité réseau**.

Critère	VPS Jux (51.77.141.54)	NAS sasnexte (82.66.244.248)
IP publique fixe	Oui	Non confirmée — SSH déjà signalé comme parfois inaccessible depuis l'extérieur (fail2ban/whitelist)

Critère	VPS Jux (51.77.141.54)	NAS sasnexte (82.66.244.248)
`network_mode: host` (recommandé Linux)	Disponible (Debian)	Non disponible sous DSM (Docker/Container Manager Synology tourne dans une couche réseau propriétaire) — nécessite le mapping de ports classique + redirection sur la box
Port forwarding	Déjà maîtrisé (nginx, certbot, ufw)	Dépend de la box du FAI, potentiellement CGNAT résidentiel
Exposition déjà en place	34 containers en prod, expérience du durcissement (incident du 9 juin 2026)	Usage actuel = uniquement syncs rclone sortants vers kDrive, pas de service entrant exposé
Impact en cas d'indisponibilité	VPS pro, uptime élevé	Dépend de l'alimentation/connexion internet du domicile

Conclusion : le VPS reste le choix le plus robuste pour un usage "accès à distance depuis n'importe où". Le NAS serait une option seulement si l'IP publique/DDNS et le port forwarding sont déjà fiables — ce qui n'est pas le cas aujourd'hui d'après le comportement SSH observé.

Prochaine étape proposée

Ajouter la stack `rustdesk` sur le VPS (Portainer, comme les autres services), avec ufw restreint aux ports 21115-21119 (TCP+UDP sur 21116) et 21118/21119 fermés tant que le client web n'est pas nécessaire.

Incident client Windows — BSOD à l'installation (2026-07-31)

Contexte : installation du client RustDesk sur le PC Windows 10 (`eliob`). L'installation s'est terminée par plusieurs écrans bleus consécutifs.

Diagnostic

- Analyse de l'Observateur d'événements (`Microsoft-Windows-WER-SystemErrorReporting`) : crash à 06:27:20 avec bugcheck `0x50` **(PAGE_FAULT_IN_NONPAGED_AREA)**, dump complet dans `C:\WINDOWS\MEMORY.DMP` (2,4 Go).
- Corrélation temporelle exacte avec l'extraction de deux composants pilotes bundlés par le client RustDesk :

    - `C:\Program Files\RustDesk\usbmmidd_v2` — pilote d'**écran virtuel USBMMIDD** (permet le contrôle à distance sans moniteur physique branché)
    - `C:\Program Files\RustDesk\drivers\RustDeskPrinterDriver` — pilote d'**imprimante virtuelle** (impression à distance)
- Vérification : ni l'un ni l'autre n'était enregistré comme périphérique actif (`Get-PnpDevice`, `pnputil /enum-drivers`) après coup → l'installation du driver a fait planter le noyau **avant** de s'enregistrer. Un problème connu et documenté côté communauté RustDesk : le driver `usbmmidd` (ancien, non WHQL sur les builds récentes de Windows 10) provoque des BSOD à l'installation sur certaines configs.
- **Point distinct identifié en passant** : la machine a un historique chronique de BSOD `0x9f` (DRIVER_POWER_STATE_FAILURE) récurrent environ une fois par mois depuis janvier 2026 (`Kernel-Power` event 41), sans lien avec RustDesk — probablement un driver qui répond mal à la mise en veille. Non résolu, à creuser séparément si besoin.

Fix appliqué

Neutralisation des deux dossiers de drivers sans désinstaller RustDesk (le contrôle à distance standard, moniteur physique branché, ne dépend pas d'eux) :

```
Stop-Service -Name "RustDesk" -Force
Stop-Process -Name "rustdesk" -Force -ErrorAction SilentlyContinue
Rename-Item "C:\Program Files\RustDesk\usbmmidd_v2" "usbmmidd_v2.disabled"
Rename-Item "C:\Program Files\RustDesk\drivers\RustDeskPrinterDriver"
"RustDeskPrinterDriver.disabled"
Start-Service -Name "RustDesk"
```

Note importante : ces commandes nécessitent une session PowerShell réellement élevée (clic droit → "Exécuter en tant qu'administrateur", accepter l'UAC). Une session non élevée renvoie la même erreur `Impossible d'ouvrir le service` aussi bien pour `Stop-Service` que pour le renommage des dossiers dans `Program Files` — ça peut faire croire à tort qu'on est admin alors que non.

Résultat

- Service `RustDesk Service` : `Running`
- Contrôle à distance classique opérationnel (vérifié : process `RustDesk.exe` actif, service démarré)
- Écran virtuel et impression à distance désactivés (fonctionnalités non utilisées) — évite que RustDesk retente d'installer ces drivers automatiquement
- **Mot de passe permanent + 2FA activés** par Julien le 2026-07-31 (menu **Sécurité** de RustDesk) → accès non attendu opérationnel : ce PC est joignable via RustDesk sans présence devant l'écran, ID + mot de passe permanent (2FA en complément).

Traversée VPS — client configuré (2026-07-31)

Le client RustDesk du PC eliob passe désormais par le serveur auto-hébergé (`hbbs`/`hbbr` sur le VPS Jux, voir plus haut) au lieu des relais publics RustDesk. Confirmé côté serveur : containers `hbbs`/`hbbr` up, ufw ouvert sur 21115-21117 (v4+v6), clé publique `id_ed25519.pub` en place. Tout le trafic de contrôle à distance de ce PC transite donc par une infrastructure maîtrisée plutôt que par un tiers.

Revision #5
Created 2026-07-30 20:07:19 UTC by Julien
Updated 2026-07-31 06:51:45 UTC by Julien